МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РЕСПУБЛИКИ КАЗАХСТАН

КАЗАХСКИЙ УНИВЕРСИТЕТ ТЕХНОЛОГИИ И БИЗНЕСА имени К.Кулажанова

Кафедра «Информационные технологии»

**Отчет**

**о научно-исследовательской работе**

**НИРМ.04**

Факультет Технологический

Образовательная программа: 7М06136 - Информационные системы

Тулегенов Алишер Канатович

Направление обучения: ғылыми және педагогикалық/научное и педагогическое

Научный руководитель Акишев Каршыга Максутович, к.т.н., асс. профессор

Астана, 2024

**СОДЕРЖАНИЕ**

**ВВЕДЕНИЕ** ....................................................................................................................... 3

1. **АВТОМАТИЗАЦИЯ ТЕХНИКИ БЕЗОПАСНОСТИ** ........................................... 5

1.1. СУОТ и их требования ............................................................................................... 5

1.2. Создание электронного журнала по технике безопасности ................................... 6

1.3. Что требуется в ПО для конкурентоспособности ................................................... 7

2. **АВТОМАТИЗАЦИЯ УПРАВЛЕНИЯ ЖУРНАЛОМ ТЕХНИЧЕСКОЙ БЕЗОПАСНОСТИ С ИСПОЛЬЗОВАНИЕМ СОВРЕМЕННЫХ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ** .................................................................. 8

2.1. Архитектура приложения .......................................................................................... 9

2.2. Структура программного обеспечения ................................................................... 11

2.3. Создание программного интерфейса ...................................................................... 15

3. **ТЕСТИРОВАНИЕ И ВНЕДРЕНИЕ ПО** ................................................................ 18

**ЗАКЛЮЧЕНИЕ** ............................................................................................................. 23

**СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ** ................................................. 24

**ВВЕДЕНИЕ**

Автоматизация процессов ведения журнала техники безопасности открывает перед предприятиями новые возможности для оптимизации и повышения эффективности своей безопасностной культуры. Разработка программного обеспечения, способного интегрировать различные аспекты управления безопасностью, от мониторинга состояния оборудования до анализа данных по происшествиям, играет ключевую роль в этом процессе.

Онлайн ведение журнала техники безопасности не только сокращает временные затраты на рутинные операции по заполнению и обработке данных, но и повышает их достоверность и актуальность. Возможность оперативного обновления информации и мгновенного доступа к ней со стороны ответственных сотрудников существенно снижает риски возникновения чрезвычайных ситуаций и повышает уровень готовности предприятия к ним.

Кроме того, цифровизация журнала техники безопасности способствует улучшению системы аналитики и прогнозирования. Современные алгоритмы машинного обучения и анализа данных позволяют выявлять скрытые закономерности и тренды, на основе которых можно разрабатывать более эффективные стратегии предотвращения происшествий и управления рисками.

Более того, переход на онлайн формат ведения журнала техники безопасности открывает новые возможности для обучения и повышения квалификации персонала. Интерактивные материалы, тесты и тренировки, доступные через цифровую платформу, делают процесс обучения более удобным и доступным, что способствует формированию культуры безопасности на предприятии.

Таким образом, автоматизация журнала техники безопасности представляет собой не просто техническое обновление, а важный шаг в направлении улучшения безопасности на предприятии в целом. Это инвестиции в будущее, направленные на повышение эффективности и надежности производственных процессов и защиту интересов как сотрудников, так и предприятия в целом.

*Актуальность*. В современном Казахстане, безопасности и охране труда на производстве уделяется большое внимание. Вместе с тем, на предприятиях до сих пор работа инженера по технике безопасности организована в традиционном стиле. Принятый в 2022 году Закон о цифровизации, требует перевода документации в цифровой формат с целью снижения бюрократии и повышения производительности труда.

*Цель работы*. Повышение эффективности труда инженера по технике безопасности предприятия.

*Задачи исследования*:

1. Анализ функциональных обязанностей инженера по технике безопасности предприятия.
2. Исследование цифровых программ автоматизации деятельности инженера по технике безопасности предприятия.
3. Выбор и обоснование инструментов для разработки автоматизированного рабочего места инженера по технике безопасности предприятия.
4. Разработка программы «Автоматизированное рабочее места инженера по технике безопасности предприятия».
5. Апробация программы «Автоматизированное рабочее места инженера по технике безопасности предприятия» на базе действующего промышленного предприятия.

*Научная новизна* заключается в разработанной методике и программа для компьютера.

*Научная и практическая значимость* исследования является повышение эффективности и производительности персонала, внедрение современных программных продуктов.

1. **АВТОМАТИЗАЦИЯ ТЕХНИКИ БЕЗОПАСНОСТИ**

*1.1 Роль автоматизации и СУОТ в обеспечении техники безопасности*

Автоматизация - это процесс внедрения современных технологий и систем для автономного выполнения задач без значительного вмешательства человека. В современном мире автоматизация играет ключевую роль, пересматривая традиционные методы работы и управления в различных отраслях. Этот процесс не только ускоряет выполнение рутинных задач, но и существенно повышает эффективность организационных процессов.

Одним из важных аспектов автоматизации является сокращение операционных затрат. За счет оптимизации процессов и уменьшения человеческого вмешательства компании могут значительно снизить расходы на обслуживание и управление. Кроме того, автоматизация способствует улучшению качества продукции или услуг, поскольку стандартизация процессов и исключение человеческих ошибок снижают вероятность дефектов.

Более того, автоматизация позволяет организациям быстрее адаптироваться к изменяющимся условиям рынка. Гибкость и адаптивность автоматизированных систем делают их незаменимым инструментом для реагирования на новые вызовы и возможности. Таким образом, внедрение автоматизации становится важным стратегическим решением для компаний, стремящихся не только оптимизировать свою деятельность в настоящем, но и обеспечить свою конкурентоспособность в будущем.

Автоматизация в журнале по технике безопасности играет ключевую роль в современном подходе к обеспечению безопасности объектов и процессов. Этот процесс внедрения передовых технологий и систем способствует эффективному контролю и управлению рисками, обеспечивая высокий уровень безопасности в различных областях.

Одним из основных преимуществ автоматизации в данной сфере является повышение скорости и точности обнаружения потенциальных угроз и инцидентов. Автоматизированные системы способны непрерывно мониторить окружающую среду и оперативно реагировать на любые нештатные ситуации, что позволяет предотвращать проблемы и минимизировать риски.

Кроме того, автоматизация позволяет сократить роль человеческого фактора в процессах контроля и управления. Это снижает вероятность человеческих ошибок и повышает надежность систем безопасности. Автоматизированные системы также способны адаптироваться к изменяющимся условиям и проводить анализ больших объемов данных для выявления аномалий, что делает их более эффективными в предотвращении угроз.

Таким образом, автоматизация в журнале по технике безопасности становится необходимым элементом в обеспечении надежной защиты объектов и снижении рисков возникновения чрезвычайных ситуаций. Это обеспечивает организациям возможность эффективного управления безопасностью и защиты как собственных сотрудников, так и важных ресурсов.

Вот некоторые ключевые причины, почему автоматизация журнала по ТБ является важным и неотъемлемым аспектом:

Эффективность и точность: Ручное ведение журнала ТБ может быть трудоемким и подверженным ошибкам человека. Автоматизация устраняет человеческий фактор, что приводит к повышению точности и эффективности сбора, анализа и документирования данных. Это особенно важно в условиях высокой загруженности персонала и больших объемов информации.

Соблюдение нормативов и стандартов: В различных отраслях существуют строгие нормативы и стандарты в области безопасности. Автоматизированный журнал ТБ способствует соблюдению этих стандартов, предоставляя надежный способ систематической регистрации и анализа данных безопасности. Это облегчает аудит и подготовку отчетов, что важно для соблюдения требований и нормативов.

Оптимизация ресурсов: Позволяет анализировать данные и выявлять области, где можно оптимизировать использование ресурсов. Анализ показателей безопасности и производственной эффективности может помочь выявить узкие места, проблемы и неэффективные практики. Это позволяет распределять ресурсы более эффективно и повышать производительность.

Улучшение культуры безопасности: Внедрение автоматизированных систем ТБ способствует формированию культуры безопасности в организации. Регулярный мониторинг и анализ данных создает более высокий стандарт безопасности, что образует осведомленный и ответственный подход персонала к соблюдению безопасных практик и процедур.

Системы Управления Оперативной Технологической информацией (СУОТ) представляют собой комплексные программные и аппаратные средства, предназначенные для сбора, хранения, обработки и анализа данных о технологических процессах в реальном времени. Эти системы используются в различных отраслях промышленности, где требуется непрерывное мониторирование и управление производственными процессами.

СУОТ включают в себя различные компоненты, такие как датчики, контроллеры, программное обеспечение для сбора и анализа данных, системы визуализации и диспетчерское оборудование. Они позволяют оперативно отслеживать параметры производственных процессов, контролировать работу оборудования, выявлять и предотвращать аварийные ситуации, а также оптимизировать эффективность производства.

Целью СУОТ является обеспечение надежного и эффективного управления технологическими процессами, повышение безопасности и качества продукции, а также сокращение времени простоя оборудования и увеличение его срока службы.

В области журнала по технике безопасности СУОТ играют важную роль, предоставляя оперативную информацию о состоянии оборудования, условиях работы и других факторах, влияющих на безопасность. Это позволяет оперативно реагировать на потенциальные угрозы и минимизировать риски возникновения чрезвычайных ситуаций.



Рисунок 1. Принципы СУОТ

Одним из важнейших аспектов в организации производственных процессов является обеспечение безопасности и здоровья работников. Для этого на предприятиях внедряются специальные меры и системы, направленные на минимизацию рисков и обеспечение соответствия нормам и правилам охраны труда.

Важным составляющих таких систем является наличие штатного специалиста по охране труда, а также эффективная система управления охраной труда (СУОТ). Ниже приведены основные мероприятия и требования, которые применяются на производственных предприятиях для обеспечения безопасности труда и здоровья работников:

Обязательность наличия штатного специалиста по охране труда. Производственные предприятия со штатной численностью до 50 сотрудников могут самостоятельно определяться с введением cистемой управления охраны труда (СУОТ). Если штат компании превышает 50 человек, наличие собственного специалиста по охране труда становится обязательным.

Обучение и инструктаж. Все сотрудники, принимаемые на рабочие места, связанные с риском для здоровья, в обязательном порядке проходят первичный инструктаж по нормативам и правилам, обеспечивающим безопасность трудовой деятельности. Периодичность повторного инструктажа для персонала составляет шесть и три месяца в зависимости от степени опасности конкретного рабочего места.

Внеплановый инструктаж или переподготовка сотрудников в специализированных учебных центрах проводятся при установке нового технологического оборудования или изменениях, касающихся условий труда. Обеспечение проведения данных мероприятий возлагается на специалиста по охране труда, который составляет соответствующие списки работников, заверяющиеся подписью руководителя предприятия.

Информационное обеспечение по охране труда. Внедрение на производстве методов СУОТ сопровождается обязательным оповещением персонала об изменениях в политике охраны труда посредством собраний, внеплановых инструктажей и размещения информации на специальных стендах. Сотрудники, чьи рабочие места связаны с риском для жизни и здоровья должны быть проинформированы в отношении своих прав на материальную компенсацию, сокращенный рабочий день и увеличенное время отдыха (руководство организовывает ознакомление сотрудников с соответствующими документами под личную роспись). Работники могут также оформлять подписку на получение периодических изданий, посвященных изменениям и нововведениям в охрану труда.

Документальное оформление. О наличии на производстве СУОТ свидетельствуют нормативные документы, в которых излагается политика предприятия в сфере охраны труда и методы, обеспечивающие защиту здоровья персонала. От правильности документального оформления во многом зависит эффективность планирования и безопасность функционирования производственных процессов, являющихся основными задачами современного менеджмента.

*1.2 Создание электронного журнала по технике безопасности*

Электронный журнал по технике безопасности - это специализированная информационная система, предназначенная для автоматизации и учета всех событий, связанных с обеспечением безопасности на производстве. Он включает в себя записи о происшествиях, инцидентах, а также предпринятых мерах по предотвращению несчастных случаев и обеспечению безопасных условий труда.

Идея создания электронного журнала по технике безопасности заключается в том, чтобы сделать процесс учета и анализа данных об обеспечении безопасности более эффективным, надежным и удобным. Такие системы позволяют быстро и точно регистрировать инциденты, проводить анализ причин и разрабатывать меры по их предотвращению.

В условиях современного производства требования к обеспечению безопасности постоянно ужесточаются, и электронные журналы становятся все более необходимыми. Они позволяют не только вести детальный учет всех происшествий, но и анализировать данные, выявлять тенденции и предотвращать возможные риски в будущем.

Введение электронного журнала по технике безопасности стало требоваться в связи с повышением требований к безопасности на производстве, а также в целях соблюдения нормативных требований и стандартов безопасности. Такие системы помогают повысить эффективность управления безопасностью, уменьшить вероятность происшествий и обеспечить безопасные условия труда для персонала.

Многие организации имеют разветвленную филиальную структуру, со множеством удаленных сотрудников. В соответствии с законодательством все они должны проходить инструктажи по охране труда. Сведения обо всех инструктажах должны заноситься в журналы учета.

Возникает вопрос соблюдения этих требований для удаленных сотрудников организации, ведь не всегда представляется возможным приехать с бумажным журналом к сотруднику, расстояния у нас в стране большие. Проблемы налицо: ведение классических бумажных журналов представляет собой довольно утомительный и скучный процесс, усложненные процедуры проверки наличия и правильности заполнения журналов в филиале. В случае ошибок приходится вносить изменения, переписывать записи, часто нужно записывать в журнал одни и те же данные (например, фамилию, дату). Подпись в журнале легко подделать, при необходимости вообще можно переписать журнал "с нуля" за ночь. В результате бумажные журналы мало того, что не обеспечивают надежный учет, так еще и загружают сотрудников рутинной работой, требуют место для хранения, создают сложности в случае необходимости получения подписи сотрудника, находящегося в другом месте. Посещение офиса удаленным сотрудником часто бывает проблематично, а иногда и просто невозможно. Выход из этой ситуации есть, если организовать ведение журналов в электронном виде. В этом поможет закон "Об электронной подписи", описывающий условия равнозначности собственноручной и электронной подписей. Никаких запретов на ведение журналов в электронном виде нет. Но самое главное, в случае с электронными журналами и современными технологиями видеоконференцсвязи может быть легко реализована технология дистанционного инструктажа.

Для организации инструктажа и учета в электронном виде необходимо: выбрать способ формирования электронной подписи (это может быть и простая подпись на базе паролей, и графическая подпись, создаваемая с помощью специальных планшетов, и квалифицированная электронная подпись, полученная в аккредитованном удостоверяющем центре); выбрать компьютер для установки серверной части системы ведения журналов в электронном виде; назначить лицо, ответственное за ведение журналов в электронном виде, и контроль времени на сервере; установить требуемое количество клиентов системы, назначить ответственных за конкретные журналы, распределить роли пользователей (чтение / запись).

Преимуществами реализации проекта по созданию электронного журнала регистрации инструктажей, стажировки и проверки знаний по охране труда: карточка по процессам и результатам (HR); автоматическое оповещение, в случае если необходимый инструктаж просрочен; отображение списка работников, которым не проведен (просрочен) необходимый инструктаж; возможность определения периодичности проведения инструктажей; распоряжение об отстранении; протокол проверки знаний; отчет по работнику / профессии.

Нормативно не установлено требование, в соответствии с которым журналы проведения инструктажей по охране труда должны вестись в бумажном виде. Следовательно, журналы проведения инструктажей по охране труда можно вести в электронном виде. Также нужно обеспечить сохранность электронных журналов проведения инструктажей по охране труда, защиту от случайного удаления или некорректных правок.

Что касается приложения для автоматизации техники безопасности, оно должно выполнять ряд ключевых функций, чтобы эффективно поддерживать безопасность и обеспечивать надлежащее управление рисками в организации. Вот некоторые из основных функций, которые такое приложение может предоставлять:

Документирование событий - приложение должно предоставлять возможность документировать все события, инциденты и меры, предпринятые для реагирования. Каждое событие должно быть четко описано, включая дату, время, тип события и сопутствующие детали. Эти записи могут служить исторической базой для анализа, отчетности и аудита, а также для обучения и разработки лучших практик.

Соблюдение стандартов и нормативов - приложение может быть настроено для автоматического соблюдения стандартов безопасности и нормативных требований. Это включает в себя создание шаблонов отчетов, документации и процедур в соответствии с требованиями безопасности и регулятивами.

*1.3 Что требуется в ПО для конкурентоспособности*

Растущая конкуренция на рынке программного обеспечения (ПО) для области техники безопасности подчеркивает важность обеспечения высокой конкурентоспособности и качества продукта. В современном мире организации стремятся к выбору ПО, которое не только эффективно решает текущие задачи, но и готово к адаптации к будущим вызовам и требованиям. Для достижения этой цели необходимо уделить внимание ряду ключевых аспектов, которые определяют успешность ПО в контексте техники безопасности.

Первоначально, ПО должно обладать высокой функциональностью и эффективностью. Это означает, что оно должно предоставлять широкий набор инструментов и функций, способных эффективно решать разнообразные задачи, связанные с обеспечением безопасности на производстве. Кроме того, для обеспечения долгосрочной эффективности, ПО должно быть готово к масштабированию и адаптации к изменяющимся потребностям и технологиям.

Важным аспектом является также надежность и безопасность программного обеспечения. Ошибки и сбои в работе ПО могут иметь серьезные последствия в области техники безопасности, поэтому необходимо обеспечить высокий уровень стабильности и защиты данных. В этом контексте тщательное тестирование и обеспечение соответствия стандартам безопасности играют ключевую роль.

Удобство использования также является важным фактором. Интуитивный интерфейс и легкость в освоении программы помогают повысить эффективность работы пользователей и снизить время на обучение новых сотрудников. Это особенно важно в условиях динамичного производственного процесса, где требуется быстрая реакция на изменения и оперативное принятие решений.

Интеграция с другими системами также необходима для обеспечения эффективной работы ПО в комплексе с другими программными и аппаратными средствами предприятия. Возможность обмена данными и совместной работы с другими системами обеспечивает максимальную эффективность использования и снижает издержки на внедрение и поддержку ПО.

Соблюдение этих принципов и требований позволяет создать программное обеспечение, которое не только соответствует текущим потребностям, но и готово к эволюции и развитию в будущем. Такое ПО становится настоящим конкурентным преимуществом для организаций, стремящихся обеспечить безопасность и эффективность своих производственных процессов.



Рисунок 2. Взаимодействие с ПО

1. **АВТОМАТИЗАЦИЯ УПРАВЛЕНИЯ ЖУРНАЛОМ ТЕХНИЧЕСКОЙ БЕЗОПАСНОСТИ С ИСПОЛЬЗОВАНИЕМ СОВРЕМЕННЫХ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ**

*2.1 Архитектура программного обеспечения*

Разработанное приложение представляет собой клиент-серверную систему, целью которой является автоматизация процессов управления журналом технической безопасности. В основе приложения лежит возможность хранения, ознакомления и редактирования статей по технике безопасности, обеспечивая удобный доступ к информации для пользователей. Администратор системы имеет возможность управлять базой данных, добавлять новых пользователей, вносить статьи и следить за прогрессом пользователей при прохождении материалов, включая тесты с ответами. Разработка приложения направлена на повышение эффективности работы специалистов в области технической безопасности и обеспечение более удобного и прозрачного взаимодействия между администраторами и пользователями.

Во время создания программного обеспечения для автоматизации журнала техники безопасности нужно обратить внимание, что существует множество технологий и инструментов. Выбор технологий зависит от целей и требований к программному обеспечению.

Создание веб-приложения имеет несколько значительных преимуществ, которые делают его предпочтительным выбором для конкуренции с другими программными обеспечениями

* Доступность: Веб-приложения доступны через веб-браузер из любого места с доступом в интернет. Это позволяет пользователям работать с приложением на различных устройствах, включая компьютеры, планшеты и смартфоны, без необходимости установки дополнительного программного обеспечения.
* Масштабируемость: Веб-приложения легче масштабируются, так как серверная инфраструктура может быть настроена для обработки большого числа пользователей без значительных изменений в клиентской части приложения.
* Удобство обновлений: Внесение изменений и обновлений в веб-приложение проще и быстрее. Новые версии могут быть развернуты на сервере, и пользователи сразу получают доступ к обновленным функциям.
* Интеграция: Веб-приложения легче интегрировать с другими онлайн-сервисами и системами, что позволяет создавать комплексные решения и максимизировать функциональность.
* Кроссплатформенность: Веб-приложения могут быть использованы на различных операционных системах (Windows, macOS, Linux) без необходимости разработки отдельных версий для каждой платформы.
* Безопасность: Доступ к данным и функциональности контролируется на серверной стороне, что может обеспечить более высокий уровень безопасности.
* Сервисная модель: Веб-приложения могут использовать подписочную или облачную модель лицензирования, что облегчает клиентам начало работы и управление расходами.
* Опыт пользователя: Современные веб-приложения могут обладать богатым интерфейсом, близким к десктопным приложениям, обеспечивая при этом простоту в использовании.

В целом, веб-приложение обеспечивает более гибкий, доступный и обновляемый способ предоставления функциональности пользователям, что делает его хорошим выбором для создания конкурентоспособного программного решения.

Архитектура "клиент-сервер" предполагает, что процессы предоставления услуг и отправки запросов разделены между разными компьютерами в сети. Каждый из этих компьютеров выполняет свои задачи независимо. В этой концепции несколько клиентских компьютеров, находящихся в удаленных местах, отправляют запросы и получают услуги от централизованной серверной машины, которую также можно назвать хост-системой.

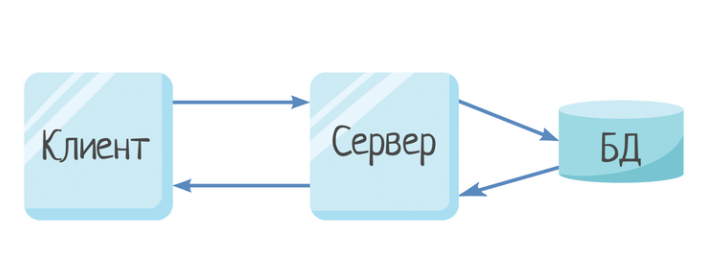


Рисунок 3. Архитектура «клиент-сервер»

Клиент-серверная архитектура была выбрана в связи с ее способностью эффективно разделять обязанности между клиентской и серверной частями системы. Преимущества такого подхода включают в себя легкость масштабирования, централизованное управление данными и возможность обеспечения высокой степени гибкости при разработке.

Функциональность клиента: Клиентская часть приложения ответственна за взаимодействие с пользователями. Это включает в себя предоставление пользовательского интерфейса, обработку пользовательских запросов и передачу данных на сервер для обработки.

Функциональность сервера: Серверная часть занимается выполнением бизнес-логики приложения, управлением базой данных, аутентификацией и авторизацией. Она обеспечивает централизованное хранение данных и координацию работы множества клиентов.

Преимущества клиент-серверной архитектуры:

- Разделение обязанностей: Клиент и сервер выполняют разные функции, что облегчает разработку, тестирование и обслуживание системы.

- Централизация данных: Централизованное хранение данных на сервере обеспечивает единый источник правды, облегчает управление данными и поддерживает их целостность.

- Масштабируемость: Гибкость масштабирования позволяет системе эффективно адаптироваться к изменяющимся нагрузкам путем добавления новых клиентов или серверов.

- Прозрачность: Взаимодействие между клиентом и сервером может быть скрыто от пользователя, обеспечивая прозрачность внутренних механизмов работы системы.

- Безопасность: Механизмы безопасности, такие как шифрование данных и аутентификация, могут быть легко реализованы и контролируются централизованно на сервере.

- Гибкость и обновляемость: Независимость клиентов и серверов позволяет легко вносить изменения в одну из частей системы без необходимости изменения другой.

Недостатки клиент-серверной архитектуры:

- Точка отказа: Если сервер выходит из строя, вся система может стать недоступной для клиентов, что делает архитектуру уязвимой к отказам сервера.

- Зависимость от сети: Работоспособность клиентов может зависеть от стабильности и пропускной способности сети, что может стать проблемой в условиях низкой связанности.

- Сложность обеспечения согласованности данных: Централизованное хранение данных может привести к сложностям в обеспечении согласованности данных при параллельных изменениях.

- Сложность масштабирования баз данных: При увеличении числа клиентов и объема данных может возникнуть необходимость в сложных мероприятиях для масштабирования базы данных.

- Производительность: Загруженность сервера может быть узким местом, особенно в случае большого числа активных клиентов, что влияет на производительность системы.

- Сложность обновлений на клиентской стороне: Изменения в логике приложения на клиентской стороне могут потребовать обновлений на каждом устройстве клиента.

Клиент-серверная архитектура предоставляет эффективный способ построения распределенных систем, однако требует внимательного управления и адаптации к конкретным требованиям и ограничениям проекта. Эффективное использование этой архитектуры требует сбалансированности между ее преимуществами и недостатками в соответствии с конкретными потребностями и целями разрабатываемой системы.

В будущем сервер-клиент архитектура продолжит играть ключевую роль в разработке программного обеспечения, поскольку она обеспечивает гибкость, масштабируемость и удобство взаимодействия между компонентами системы. С постоянным развитием технологий и изменением требований рынка, можно ожидать следующих тенденций в развитии этой архитектуры:

Рост микросервисной архитектуры: С развитием микросервисной архитектуры, сервер-клиент взаимодействие станет более модульным и децентрализованным. Компании будут предпочитать разбивать свои системы на небольшие, автономные сервисы, что способствует более гибкой разработке, развертыванию и масштабированию приложений.

Развитие протоколов и стандартов: Стандарты и протоколы взаимодействия между сервером и клиентом будут постоянно совершенствоваться. Это включает в себя улучшение протоколов передачи данных, таких как HTTP/2 и HTTP/3, а также стандартизацию протоколов авторизации и аутентификации, обеспечивая повышенную безопасность и производительность.

Увеличение использования облачных вычислений: С ростом облачных технологий и услуг, сервер-клиент архитектура будет все больше ориентироваться на облачные вычисления. Это обеспечит более высокую отказоустойчивость, масштабируемость и доступность приложений за счет использования облачных платформ для размещения серверных компонентов.

Использование новых технологий: С развитием технологий, таких как искусственный интеллект, интернет вещей (IoT) и блокчейн, сервер-клиент архитектура будет интегрировать эти инновации для создания более интеллектуальных и ресурсоэффективных приложений. Например, использование алгоритмов машинного обучения на сервере для анализа данных клиентов или интеграция с устройствами IoT для сбора и обработки данных в реальном времени.

Улучшение пользовательского опыта: С развитием веб-технологий, таких как веб-сокеты и серверные события, а также ростом мобильных приложений, сервер-клиент архитектура будет все более фокусироваться на обеспечении высокопроизводительного и реактивного пользовательского опыта. Это включает в себя использование техник кэширования данных на клиентской стороне, предварительной загрузки и отложенной загрузки ресурсов для ускорения работы приложений.

В целом, сервер-клиент архитектура будет продолжать эволюционировать, чтобы соответствовать требованиям современного программного обеспечения, обеспечивая высокую производительность, безопасность и масштабируемость приложений в различных сферах применения.

*2.2 Структура программного обеспечения*

Взаимодействие между клиентом и сервером в клиент-серверной архитектуре осуществляется с использованием различных протоколов и технологий, которые играют ключевую роль в обмене данными и обеспечивают эффективную коммуникацию между компонентами системы.

HTTP, как протокол прикладного уровня, служит стандартным средством обмена данными в веб-приложениях. Он основан на концепции "запрос-ответ", где клиент инициирует запрос на сервер, а сервер отвечает передачей данных обратно клиенту. Этот протокол широко применяется для загрузки веб-страниц, изображений и других ресурсов, а также для создания RESTful API, предоставляя стандартизированный интерфейс для взаимодействия.

WebSocket представляет собой технологию, обеспечивающую полнодуплексное взаимодействие между клиентом и сервером. Его особенность заключается в возможности эффективного обмена данными в режиме реального времени. Одно установленное соединение позволяет отправлять данные как от клиента к серверу, так и наоборот. WebSocket часто используется в разработке чатов, мессенджеров и других приложений, где требуется мгновенная передача информации.

RESTful API представляет собой подход к взаимодействию между клиентом и сервером, основанный на представлении данных в виде ресурсов с уникальными идентификаторами (URI). Этот подход поддерживает ограниченный интерфейс и четко определенные методы запросов, такие как GET, POST, PUT, DELETE. RESTful API широко применяется для создания веб-сервисов и взаимодействия с данными в веб-приложениях.

Node.js - это среда выполнения JavaScript, основанная на движке V8 Chrome, которая позволяет выполнять JavaScript на серверной стороне приложения. Одним из главных преимуществ Node.js является его асинхронная и событийно-ориентированная архитектура, которая позволяет эффективно обрабатывать большие объемы запросов без блокировки потоков, что обеспечивает высокую производительность и отзывчивость приложения.

Npm (Node Package Manager) - это основной инструмент для управления зависимостями и пакетами в экосистеме Node.js. Он предоставляет доступ к огромному количеству пакетов, которые разработаны сообществом Node.js, и позволяет легко управлять зависимостями в ваших проектах.

Основные возможности npm включают:

- Установка и обновление пакетов: npm позволяет устанавливать сторонние пакеты и обновлять их до последних версий с помощью простых команд в терминале.

- Управление зависимостями: Вы можете определить зависимости вашего проекта в файле package.json, который затем используется для установки всех необходимых пакетов вместе с их версиями.

- Публикация собственных пакетов: Разработчики могут публиковать свои собственные пакеты в реестр npm, делиться своим кодом с сообществом и использовать его в других проектах.

- Управление версиями: npm предоставляет механизмы для управления версиями пакетов, включая возможность установки конкретной версии или ограничение диапазона версий для определенной зависимости.

- Интеграция с CI/CD: npm позволяет интегрировать управление зависимостями в ваши процессы непрерывной интеграции и развертывания, что обеспечивает стабильность и надежность вашего приложения.

В сочетании с фреймворком Express.js, Node.js становится мощным инструментом для разработки веб-приложений. Express.js предоставляет удобный и гибкий способ создания веб-серверов и обработки HTTP запросов. С его помощью можно легко определять маршруты, обрабатывать запросы, управлять сессиями и многое другое. Это упрощает разработку веб-приложений и позволяет сосредоточиться на бизнес-логике приложения, минимизируя издержки на инфраструктурные задачи.

Одним из ключевых преимуществ использования Node.js и Express.js является возможность разработки приложений на JavaScript как на клиентской, так и на серверной стороне. Это упрощает обмен кодом и легко масштабируется, что ускоряет разработку и упрощает поддержку приложения.

Кроме того, использование Node.js и Express.js обеспечивает нам большую гибкость и возможность интеграции с различными сторонними сервисами и библиотеками благодаря обширной экосистеме Node.js. Мы можем легко использовать тысячи пакетов npm для добавления дополнительной функциональности, такой как аутентификация, авторизация, работа с базами данных и многое другое, что позволяет нам быстро расширять возможности нашего приложения и следовать лучшим практикам разработки.

Таким образом, выбор Node.js и Express.js для серверной стороны нашего приложения обеспечивает нам высокую производительность, гибкость, простоту разработки и расширения, что позволяет нам эффективно создавать и поддерживать мощные веб-приложения.

MongoDB - это NoSQL база данных, которую мы выбрали для нашего приложения, интегрированную через Mongoose. Она отличается гибкостью в хранении и обработке данных. В отличие от традиционных реляционных баз данных, MongoDB использует гибкие JSON-подобные документы для хранения данных, что позволяет нам эффективно моделировать сложные структуры данных и легко изменять их по мере развития приложения.

Интеграция с MongoDB через Mongoose обеспечивает нам удобный и выразительный способ взаимодействия с базой данных. Mongoose - это объектно-документный отображатель (ODM) для MongoDB, который предоставляет нам возможность определять схемы данных, создавать модели и выполнять запросы к базе данных с помощью простого и понятного API. Это упрощает разработку и поддержку базы данных, а также обеспечивает защиту от ошибок и нежелательных изменений данных.

Выбор MongoDB и Mongoose обеспечивает нам гибкость в работе с данными. Мы можем легко изменять структуру данных и добавлять новые поля или коллекции по мере необходимости, что позволяет нам быстро реагировать на изменения требований бизнеса или пользовательских запросов.

Кроме того, MongoDB хорошо масштабируется горизонтально, что означает, что мы можем легко увеличивать производительность и доступность базы данных путем добавления новых узлов кластера MongoDB. Это позволяет нам обеспечивать высокую производительность и надежность нашего приложения даже при увеличении его нагрузки или объема данных.

Таким образом, выбор MongoDB и Mongoose для нашего приложения обеспечивает нам гибкость, простоту разработки и масштабируемость базы данных, что позволяет нам эффективно управлять данными и обеспечивать высокую производительность нашего приложения.

На клиентской стороне приложения мы используем ванильный JavaScript, HTML (включая EJS для серверного рендеринга) и CSS. Этот стек технологий обеспечивает удобное взаимодействие пользователя с приложением, а также обеспечивает легкость разработки и поддержки интерфейса. JavaScript является основным языком программирования для взаимодействия с пользователем и обработки динамического контента на стороне клиента. HTML используется для структурирования содержимого страницы, а CSS - для оформления и стилизации элементов интерфейса.

Использование EJS для серверного рендеринга HTML позволяет нам создавать динамические страницы на стороне сервера, что улучшает производительность и обеспечивает более быстрое отображение контента для пользователей. Это также упрощает поддержку и обновление интерфейса, так как изменения могут быть внесены централизованно на сервере без необходимости внесения изменений в каждую страницу вручную.

В результате мы получаем мощный и эффективный стек технологий, который позволяет нам создавать пользовательский интерфейс с высокой степенью интерактивности и отзывчивости. Это способствует повышению удовлетворенности пользователей и обеспечивает более плавный и приятный опыт использования нашего приложения.

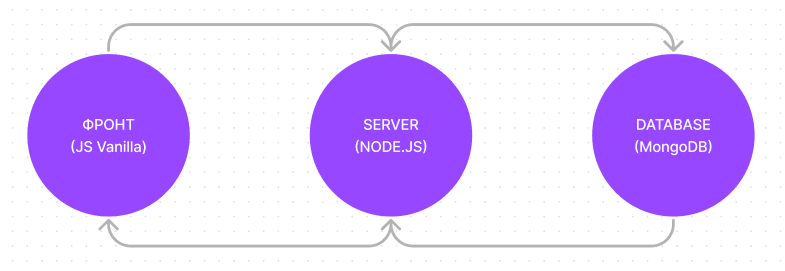


Рисунок 4. Использованный стек (архитектурный)

Проект включает использование GitHub для эффективного управления версиями кода и обеспечивает удобное отслеживание изменений. GitHub - это популярная платформа для хостинга и совместной работы над проектами с использованием системы контроля версий Git. Наше использование GitHub позволяет нам:

Управлять версиями кода: Мы можем легко создавать новые версии кода, фиксировать изменения и отслеживать историю изменений с помощью системы контроля версий Git. Это обеспечивает нам контроль над эволюцией нашего приложения и упрощает возврат к предыдущим версиям кода в случае необходимости.

Совместную разработку: GitHub предоставляет инструменты для совместной работы над проектом несколькими разработчиками. Мы можем легко создавать ветки для новых функций или исправлений ошибок, проводить обзоры кода и слияние изменений, что способствует эффективной и организованной разработке.

Отслеживание задач и ошибок: Мы можем использовать GitHub Issues для управления задачами, отслеживания ошибок и запросов на улучшение функционала. Это обеспечивает централизованное место для общения и совместной работы по улучшению нашего приложения.

Документация и вики-страницы: GitHub позволяет нам создавать и поддерживать документацию и вики-страницы для нашего проекта. Это помогает нам документировать процессы, инструкции по установке и использованию приложения, а также предоставлять полезную информацию для участников проекта и конечных пользователей.

Интеграция с CI/CD: Мы можем интегрировать GitHub с системами непрерывной интеграции и доставки (CI/CD), чтобы автоматизировать процессы сборки, тестирования и развертывания приложения. Это помогает нам обеспечивать высокое качество кода и быструю поставку изменений в продуктивную среду.

Таким образом, использование GitHub обеспечивает нам удобный и эффективный способ управления версиями кода, совместной работы над проектом и отслеживания изменений, что способствует повышению производительности и качества нашего приложения.

*Роли в системе:*

Администратор (admin): Обладает полным доступом к системе. Может создавать новых пользователей, добавлять статьи по технике безопасности, а также управлять базой данных. Авторизация администратора происходит через страницу входа.

Пользователь (user): Имеет доступ к чтению статей, прохождению тестов и взаимодействию с материалами в системе. Для безопасности и предотвращения несанкционированных аккаунтов, создание новых пользователей осуществляется только администратором.

*Структура базы данных:*

Таблица "articles": Хранит статьи по технике безопасности, их содержание и дополнительные атрибуты.

Таблица "users": Содержит информацию о пользователях, их учетные данные и статус активации.

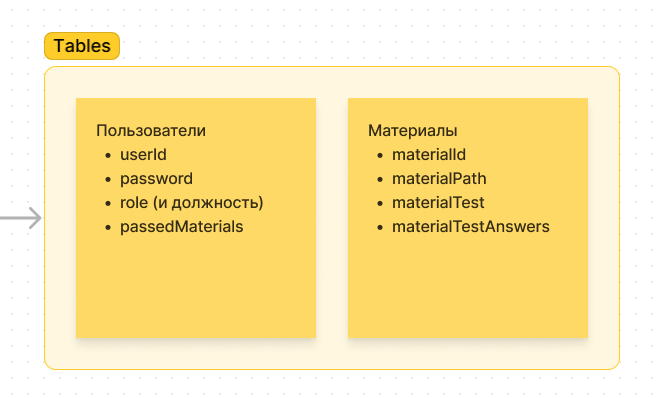


Рисунок 5. Изначальная структура БД

*Процесс авторизации и безопасности:*

* Пользователи вводят свои учетные данные на странице авторизации (логин и пароль).
* Создание новых аккаунтов осуществляется исключительно администратором для обеспечения безопасности.
* После создания аккаунта пользователю необходимо активировать его, следуя инструкциям, отправленным на указанный им адрес электронной почты.
* Это обеспечивает защиту от нежелательных аккаунтов и уменьшает риск DDoS-атак.

*2.3 Создание программного интерфейса*

В процессе создания первоначального дизайна программы был применён инструмент - Figma. На нем были разработаны интерфейс и проработано удобство использования программного обеспечения.

Figma - это универсальный инструмент для дизайна пользовательского интерфейса (UI), который позволяет создавать макеты, прототипы и интерфейсы в реальном времени. Он обладает широким спектром функций, включая возможность коллаборации, что позволяет командам совместно работать над проектами, делиться идеями и мгновенно получать обратную связь. Благодаря простому интерфейсу и доступности через веб-браузер, Figma становится удобным и эффективным инструментом для дизайнеров и разработчиков, независимо от их местоположения и устройства, на котором они работают.

Одним из ключевых преимуществ Figma является его гибкость и масштабируемость. Инструмент обеспечивает возможность создания библиотек компонентов, что позволяет повторно использовать элементы дизайна и обеспечивает единообразие интерфейса. Кроме того, Figma поддерживает интеграцию с другими популярными инструментами и сервисами, что упрощает процесс работы и позволяет создавать комплексные решения для различных проектов.

Процесс разработки макета включал в себя 3 этапа:

- Дизайн модели - идеи того как будет выглядеть программа

- Навигация - переходы между страницами

- Работа над эргономикой - поиск лучшего размещения элементов управления и прочего контента

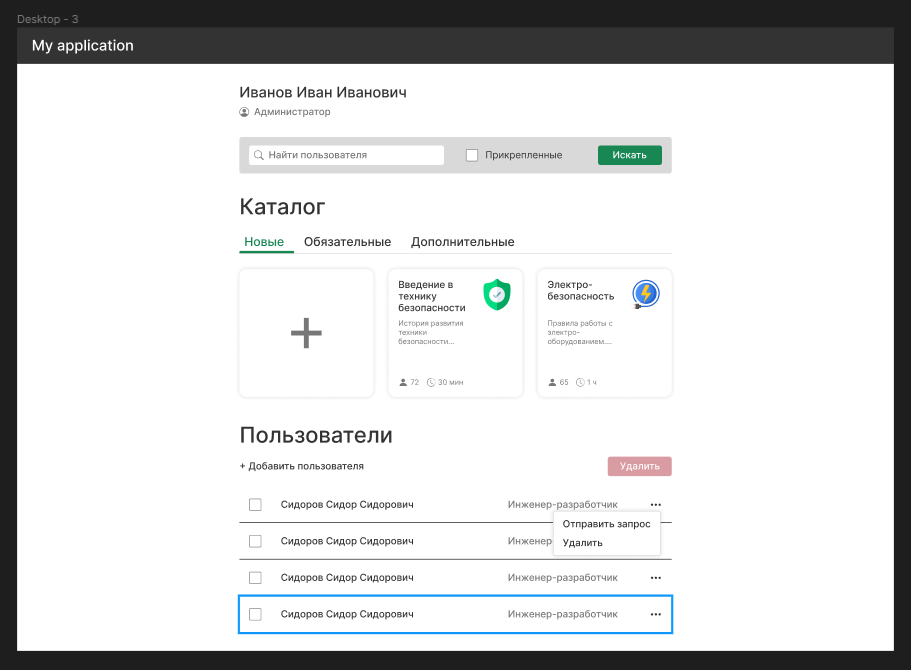


Рисунок 6. Дизайн главной страницы сайта

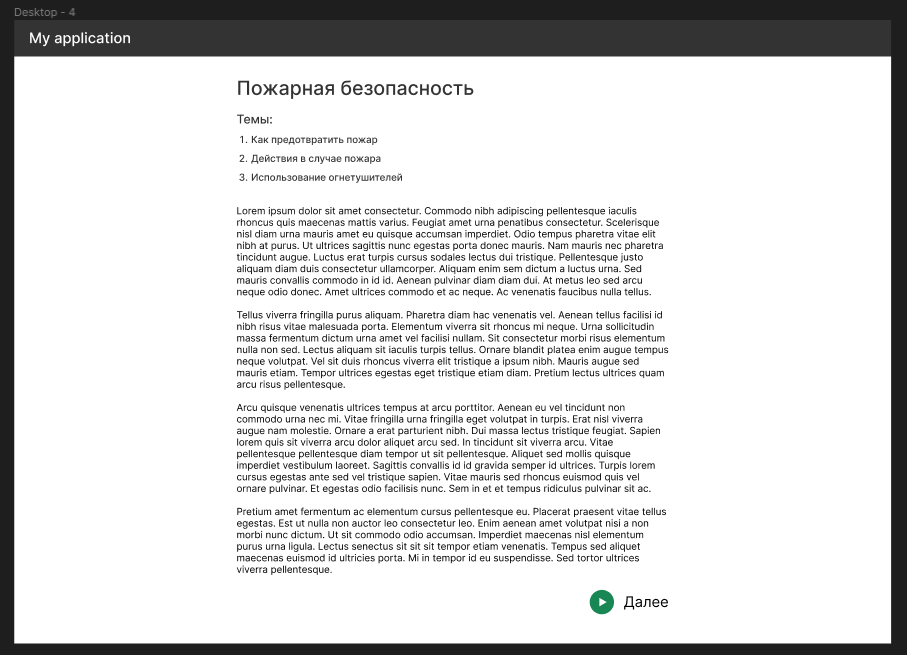


Рисунок 7. Дизайн страницы статьи

Проектирование пользовательского интерфейса в Figma предоставило обширные возможности для визуализации и оптимизации пользовательского опыта. После завершения этапа дизайна, перешел к разработке программного обеспечения в интегрированной среде разработки Visual Studio Code.

Visual Studio Code (VS Code) — это мощное средство для разработки программного обеспечения, которое обладает широким спектром функций и возможностей. Он предоставляет удобное и интуитивно понятное окружение для написания кода, поддерживая различные языки программирования и фреймворки. Одним из ключевых преимуществ VS Code является его легкая настройка и расширяемость: разработчики могут легко настроить редактор под свои нужды, выбрав необходимые плагины и расширения из обширного каталога.

Кроме того, VS Code обеспечивает удобную интеграцию с системами контроля версий, что позволяет разработчикам эффективно управлять изменениями в своем коде и сотрудничать над проектами в команде. Он также предоставляет богатый набор инструментов для отладки кода, автодополнения, анализа кода и многих других функций, что упрощает и ускоряет процесс разработки.

Выбор VS Code для разработки программного обеспечения обоснован его простотой использования, богатым функционалом и активным сообществом пользователей и разработчиков, которые постоянно работают над его улучшением и расширением возможностей. Благодаря этим преимуществам, VS Code остается одним из наиболее популярных и предпочтительных инструментов среди разработчиков по всему миру.

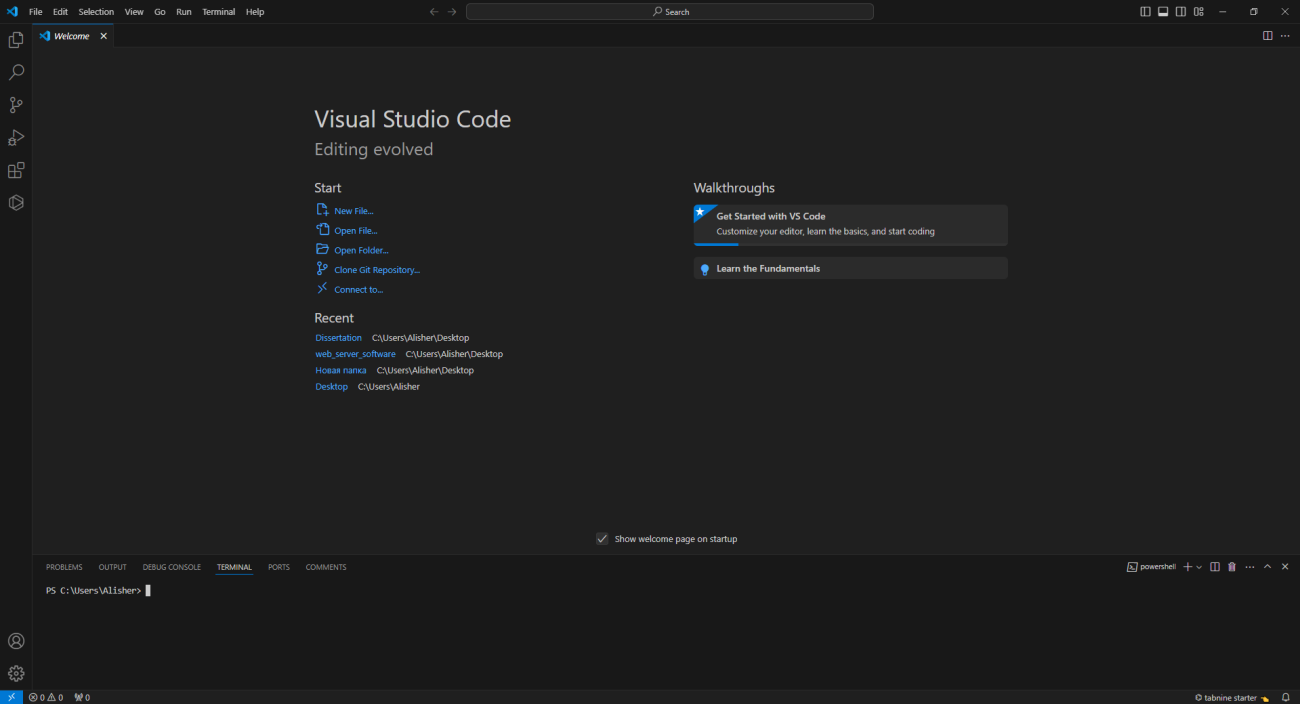


Рисунок 8. Visual Studio Code

В процессе разработки программного обеспечения были использованы следующие расширения для редактора кода Visual Studio Code:

Prettier: Это расширение позволяет автоматически форматировать код в соответствии с заданными стандартами оформления. Благодаря этому инструменту удается поддерживать единый стиль кода в проекте и повысить его читаемость.

Tabnine: Это расширение предоставляет автодополнение с использованием искусственного интеллекта. Оно предлагает контекстно-зависимые предложения и подсказки при написании кода, что значительно ускоряет процесс разработки и снижает количество ошибок.

Auto Rename Tag: Данное расширение автоматически переименовывает закрывающие теги HTML и XML при изменении открывающего тега. Это упрощает редактирование кода и сокращает время, необходимое на внесение изменений.

Russian Language: Это расширение добавляет поддержку русского языка в среду разработки VS Code, что облегчает взаимодействие с редактором для русскоязычных разработчиков и делает использование более комфортным.

Благодаря этим расширениям удалось улучшить процесс разработки, сделать его более эффективным и продуктивным.

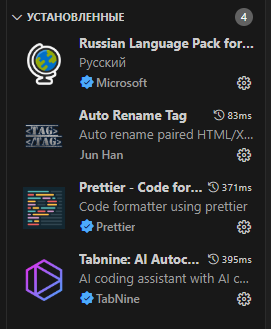


Рисунок 9. Использованные расширения

Разработка серверной части:

Для разработки ПО, сначала нужно установить Node.js - среду выполнения JavaScript. Node.js можно загрузить с официального сайта nodejs.org.

Среда выполнения (runtime environment) - это программное обеспечение, которое предоставляет необходимые ресурсы и условия для выполнения программного кода. В случае Node.js, это среда, в которой JavaScript-приложения могут выполняться на сервере.

Среда выполнения также обеспечивает доступ к ресурсам компьютера или сервера, управляет памятью, предоставляет асинхронное выполнение кода, и позволяет использовать различные библиотеки и модули для расширения функциональности приложения.

При установке Node.js с ним автоматически устанавливается npm (Node Package Manager). Npm является стандартным менеджером пакетов для JavaScript и Node.js. Он предоставляет доступ к огромному количеству библиотек, модулей и инструментов, разработанных сообществом разработчиков. Вот несколько причин, почему npm необходим для разработки на Node.js:

Управление зависимостями: npm позволяет управлять зависимостями вашего проекта, устанавливать новые пакеты, обновлять существующие и удалить те, которые больше не нужны. Это делает процесс разработки более удобным и эффективным.

После установки Node.js и npm можно проверить их наличие и версии, выполнив следующие команды в терминале или командной строке: node -v и npm -v.

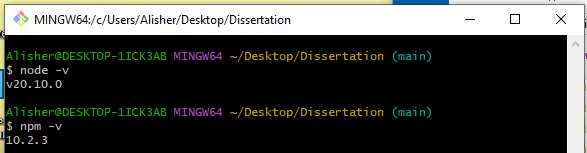


Рисунок 10. Проверка версий node и npm

Выполнив эти команды в терминале или командной строке сразу после установки Node.js и npm, можно убедиться, что они были успешно установлены и готовы к использованию. Если команды успешно выполнены, можно увидеть версии установленных программных компонентов.

Для создания серверной части приложения использовался фреймворк Express.js для Node.js. Express.js предоставляет мощные инструменты для создания веб-приложений и API на основе Node.js, делая процесс разработки быстрым и эффективным.

Для начала установки Express.js необходимо иметь уже установленные Node.js и npm. Установка Express.js выполняется с помощью npm команды:

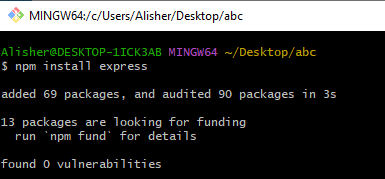


Рисунок 11. Установка Express.js

После установки Express.js необходимо создать файл сервера, в котором будет инициализирован и сконфигурирован Express. В этом файле необходимо подключить Express, создать экземпляр приложения и настроить обработчики маршрутов и промежуточное программное обеспечение.

Настройка сервера также может включать в себя создание и подключение базы данных, установку средств аутентификации и авторизации, а также настройку мониторинга и журналирования для отслеживания работы приложения.

Создание сервера с использованием Express.js обеспечивает быстрое развертывание и масштабирование веб-приложений, а также упрощает процесс обработки запросов и взаимодействия с клиентской частью приложения.

Авторизация и аутентификация:

Для создания аутентификации и авторизации в нашем Express.js приложении мы использовали модуль express-session. Этот модуль предоставляет удобный способ управления сессиями пользователей в нашем веб-приложении. Установка express-session осуществляется с помощью npm:

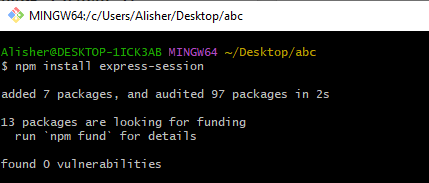


Рисунок 12. Установка модуля express-session

Основные плюсы использования express-session включают в себя:

Простота установки и использования: Установка express-session осуществляется несложным образом, и он легко интегрируется в наше Express.js приложение. Его простой API позволяет быстро добавить сессионное хранилище и управлять сессиями.

Гибкие настройки: Модуль express-session предоставляет множество настраиваемых параметров, позволяя настроить поведение сессий в соответствии с потребностями нашего приложения. Мы можем задать секретный ключ для подписи сессий, определить, должны ли сессии сохраняться при каждом запросе, и настроить другие параметры.

Поддержка различных типов хранилищ: Express-session поддерживает различные типы хранилищ сессий, такие как в памяти, на диске, в базе данных или даже внешнем хранилище типа Redis. Это позволяет выбрать наиболее подходящий тип хранилища с учетом требований и масштаба нашего приложения.

Безопасность: Модуль express-session обеспечивает безопасное хранение и передачу данных сессий, включая защиту от атак типа CSRF (межсайтовая подделка запроса). Сессионные данные хранятся на сервере, а идентификатор сессии передается клиенту в виде куки, обеспечивая защиту от утечек информации.

Интеграция с другими модулями: Express-session легко интегрируется с другими модулями и инструментами для реализации дополнительных функций, таких как аутентификация с использованием Passport.js или обработка ошибок аутентификации.

Таким образом, использование модуля express-session позволяет нам эффективно добавить аутентификацию и авторизацию в наше Express.js приложение, обеспечивая безопасное хранение сессий и удобное управление пользователями.

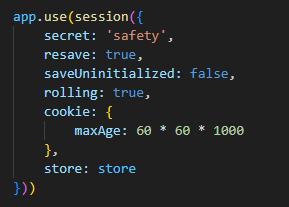


Рисунок 13. Конфиг для сессий

Для добавления базы данных в наше Express.js приложение мы использовали Mongoose - библиотеку для работы с MongoDB, которая предоставляет удобный способ создания моделей данных и выполнения запросов к базе данных.

Установка Mongoose с помощью npm:

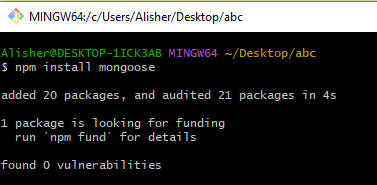


Рисунок 14. Установка модуля для использования БД MongoDB

Для создания базы данных мы воспользовались официальным сайтом MongoDB, где зарегистрировали аккаунт и получили доступ к сервису MongoDB Atlas - популярному облачному сервису управления базами данных. На MongoDB Atlas мы создали кластер, выбрав параметры, которые соответствуют нашим потребностям, такие как регион размещения серверов, размер и тип экземпляров. После создания кластера мы настроили базу данных, выбрав тип хранилища, параметры безопасности и другие настройки в соответствии с требованиями нашего проекта.

После успешной настройки кластера и базы данных мы создали необходимые коллекции, которые являются эквивалентом "таблиц" в реляционных базах данных. В нашем случае, мы создали коллекции с именами Articles и Users, которые будут хранить соответствующие данные для нашего приложения.

Затем мы получили строку подключения (Connection String), которая содержит информацию о хосте, порте, имени базы данных и учетных данных. Впоследствии использовав эту строку для подключения к нашей базе данных из приложения.

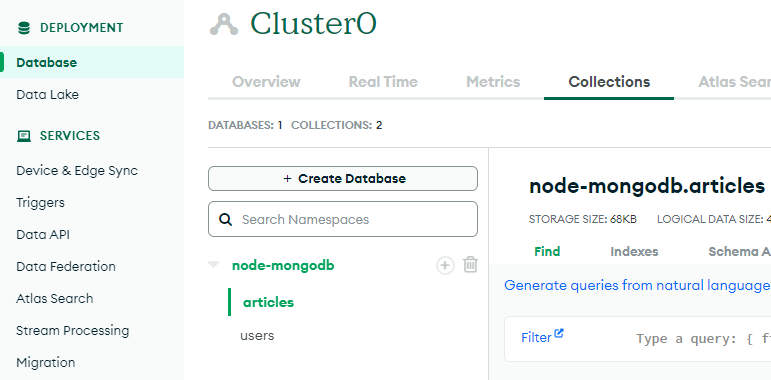


Рисунок 15. Создание таблиц в базе данных

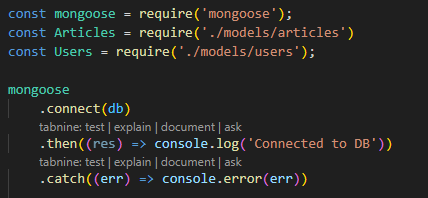


Рисунок 16. Подключение к базе данных

Здесь происходит импорт модуля Mongoose и моделей данных, а затем устанавливается подключение к базе данных MongoDB.

В следующем блоке кода представлены модель данных users, который определяют структуру и типы данных для хранения информации в базе данных MongoDB.

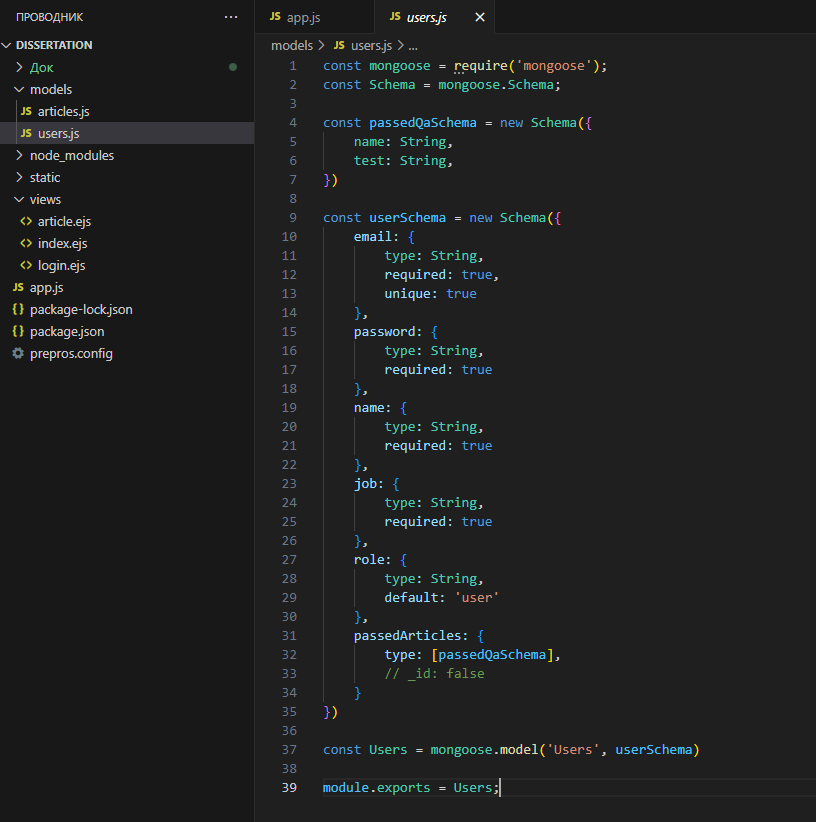


Рисунок 17. Модель данных users

Перед входом в защищенные части приложения мы используем middleware, который проверяет, авторизован ли пользователь. Middleware - это функция, которая выполняется перед обработкой запроса. В данном случае, middleware проверяет наличие у пользователя аутентификационных данных (например, сессионных данных) и определяет, имеет ли он доступ к защищенным ресурсам.

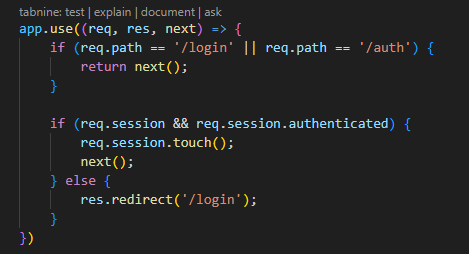


Рисунок 18. Middleware авторизованности

Далее представлен код, реализующий механизм аутентификации и авторизации пользователей в нашем приложении.

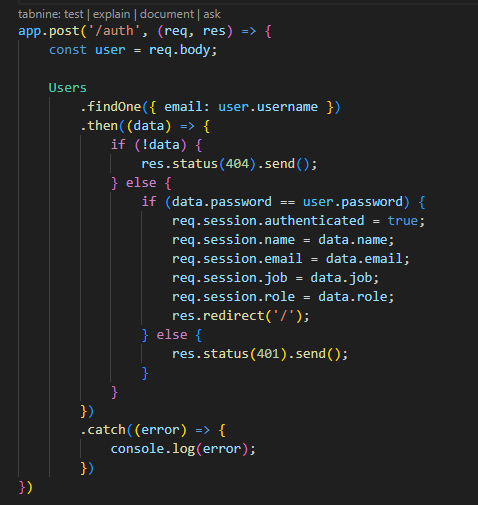


Рисунок 19. Код авторизации

Ниже приведен код маршрута на главную страницу, где используется функция render для отображения шаблона EJS. В этой функции передаются параметры, которые будут использоваться для проверки в файлах EJS и динамического отображения контента на странице.

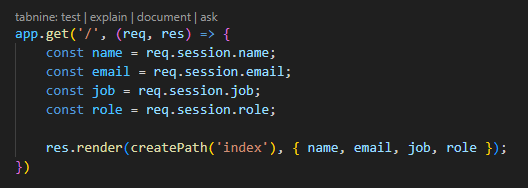


Рисунок 20. Маршрутизация на главную страницу

Для отображения динамического контента на главной странице нашего приложения мы используем EJS (Embedded JavaScript), который позволяет встраивать JavaScript код прямо в HTML файлы и создавать шаблоны для веб-страниц. Переданные ранее параметры затем доступны в шаблоне EJS для динамического формирования контента. Например, параметр role может быть использован для генерирования элементов администратора или другого контента в зависимости от роли пользователя. Например отображения блока для создание и удаления других пользователей.

Установка EJS из npm:

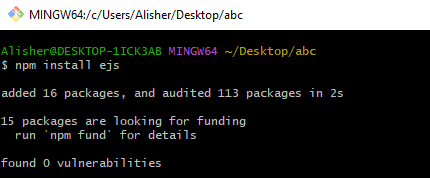


Рисунок 21. Установка модуля EJS

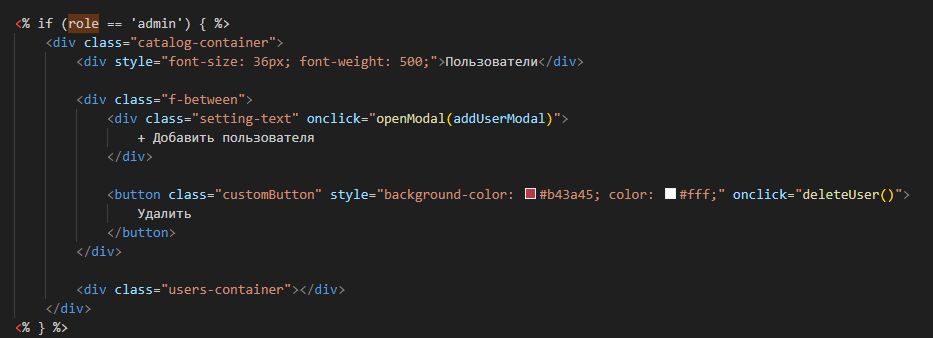


Рисунок 22. Блок проверки роли

Разработка клиентского приложения:

Для начала разработки клиентской части приложения я приступил к верстке ранее разработанного дизайна. В разметке HTML были использованы семантические теги для создания структуры страниц, а также встраивание шаблонов EJS для динамического формирования контента. Я начал с размещения основных элементов интерфейса, таких как заголовки, формы, кнопки и другие, используя теги EJS для вставки данных, которые будут заполнены на сервере.

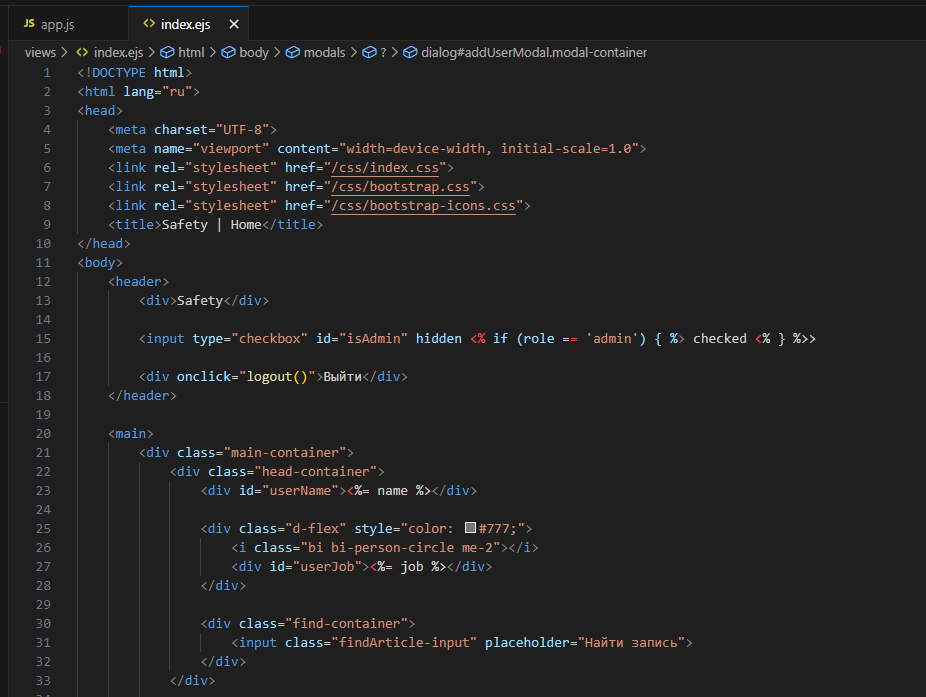


Рисунок 23. Разметка HTML + EJS

Затем приступил к стилизации страниц с помощью CSS, определяя внешний вид и расположение элементов. Для обеспечения адаптивности интерфейса я использовал медиа-запросы и другие техники CSS, которые позволяют корректно отображать страницы на различных устройствах и разрешениях экранов.



Рисунок 24. Стили элементов

Bootstrap, как фреймворк для веб-разработки, был включен в процесс разработки с целью улучшения отзывчивости и обеспечения единообразного дизайна на различных устройствах. Использование компонентов Bootstrap значительно ускорило процесс стилизации элементов интерфейса, так как позволило избежать написания множества кастомных стилей с нуля. Благодаря этому удалось значительно сократить время разработки, а также обеспечить более простую поддержку и масштабируемость проекта.

Кроме того, Bootstrap предоставляет широкий выбор готовых компонентов, таких как навигационные меню, карточки, модальные окна и многое другое, что позволяет быстро и легко создавать функциональные элементы интерфейса. Это особенно полезно в условиях ограниченных сроков разработки и требований к современному и адаптивному дизайну. Таким образом, использование Bootstrap стало неотъемлемой частью процесса разработки, обеспечивая удобство, эффективность и качество конечного продукта.

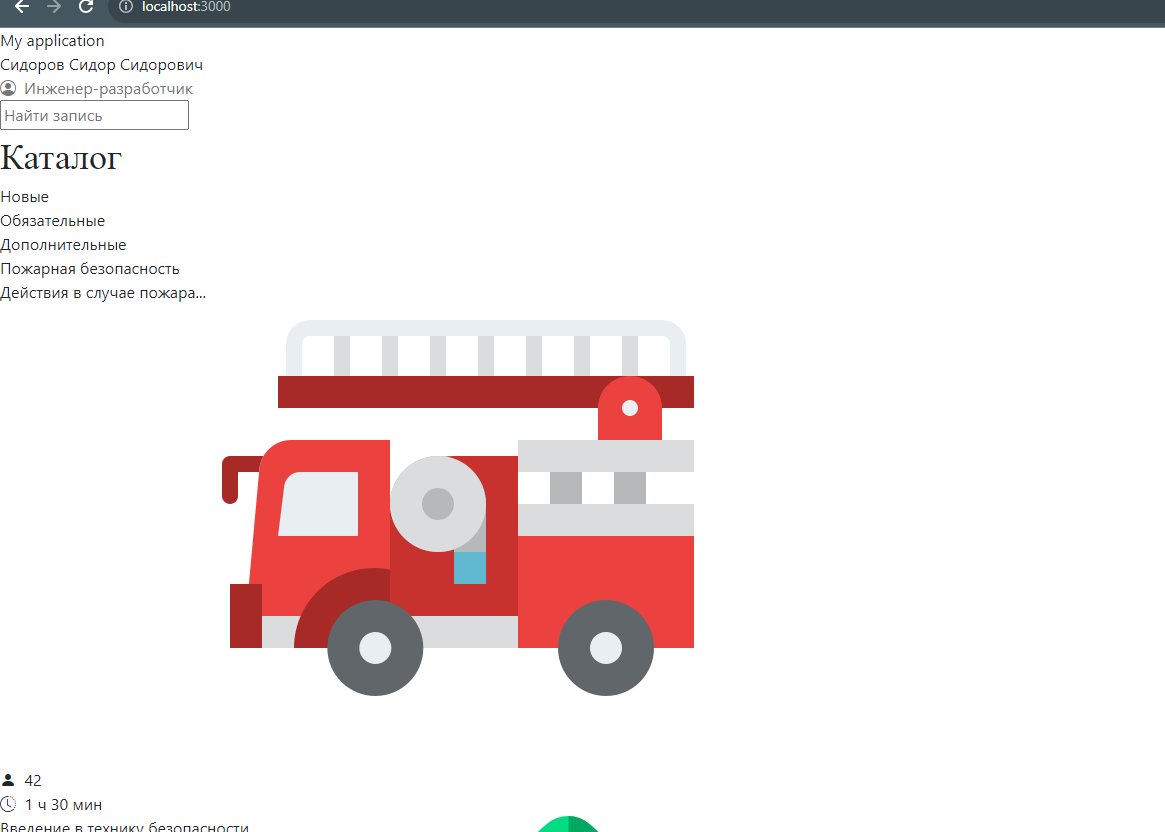


Рисунок 25. Страница без CSS и Bootstrap

После, был добавлен JavaScript (vanilla) для дальнейшего улучшения функциональности веб-приложения. JavaScript позволяет обрабатывать события пользователя, валидировать данные на стороне клиента и взаимодействовать с сервером без перезагрузки страницы, что существенно расширяет возможности приложения.

Использование JavaScript также способствует созданию более динамичного пользовательского интерфейса, позволяя добавлять анимации, изменять содержимое страницы и реагировать на действия пользователя в реальном времени. Этот шаг не только повышает функциональность приложения, но и создает более удобный и привлекательный пользовательский опыт, что является важным аспектом разработки современных веб-приложений.

JavaScript также предоставляет удобные инструменты для отладки кода прямо в браузере, что упрощает процесс разработки и исправления ошибок. Браузерные инструменты разработчика, такие как инспектор элементов, консоль JavaScript, и отладчик, позволяют разработчикам исследовать структуру DOM, выводить отладочные сообщения и отслеживать выполнение кода на лету. Благодаря этим инструментам, разработчики могут эффективно идентифицировать проблемы в своем JavaScript коде, проверять значения переменных, а также отлавливать и исправлять ошибки в реальном времени, что способствует улучшению качества кода и повышению производительности веб-приложения.

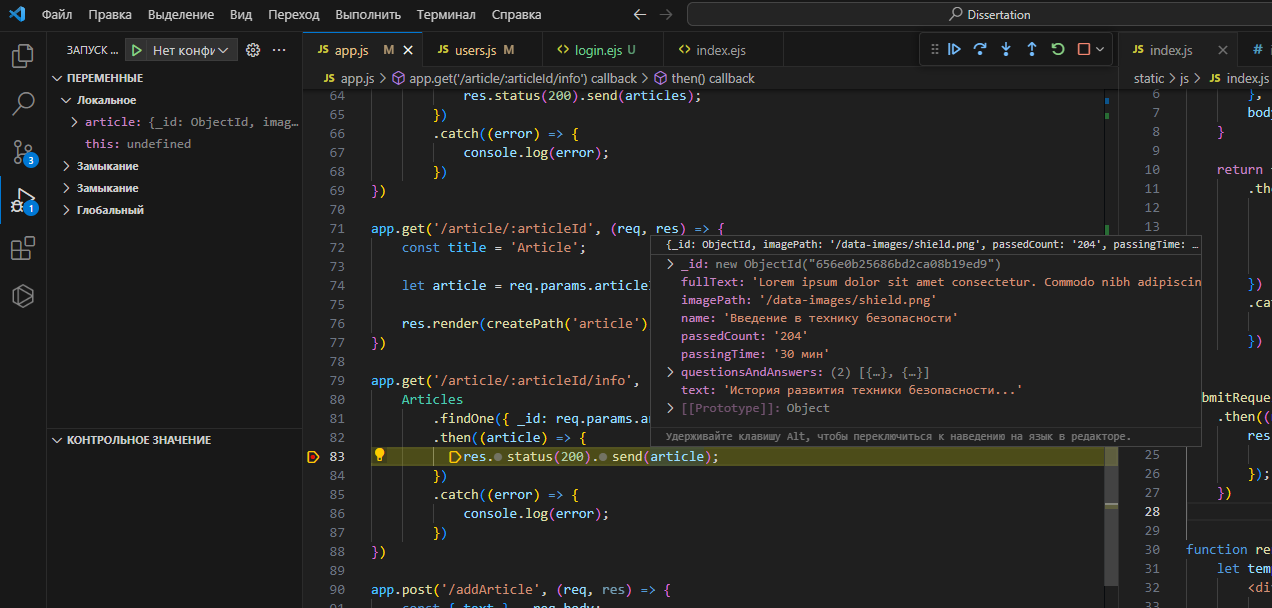


Рисунок 26. Отладка функционала

Для эффективного управления версиями в процессе разработки была внедрена система контроля версий Git. Инструмент Git позволил отслеживать изменения в коде, создавать и сливать ветки, а также управлять кодовой базой проекта. В ходе разработки Git был активно использован для фиксации изменений и ведения истории разработки.

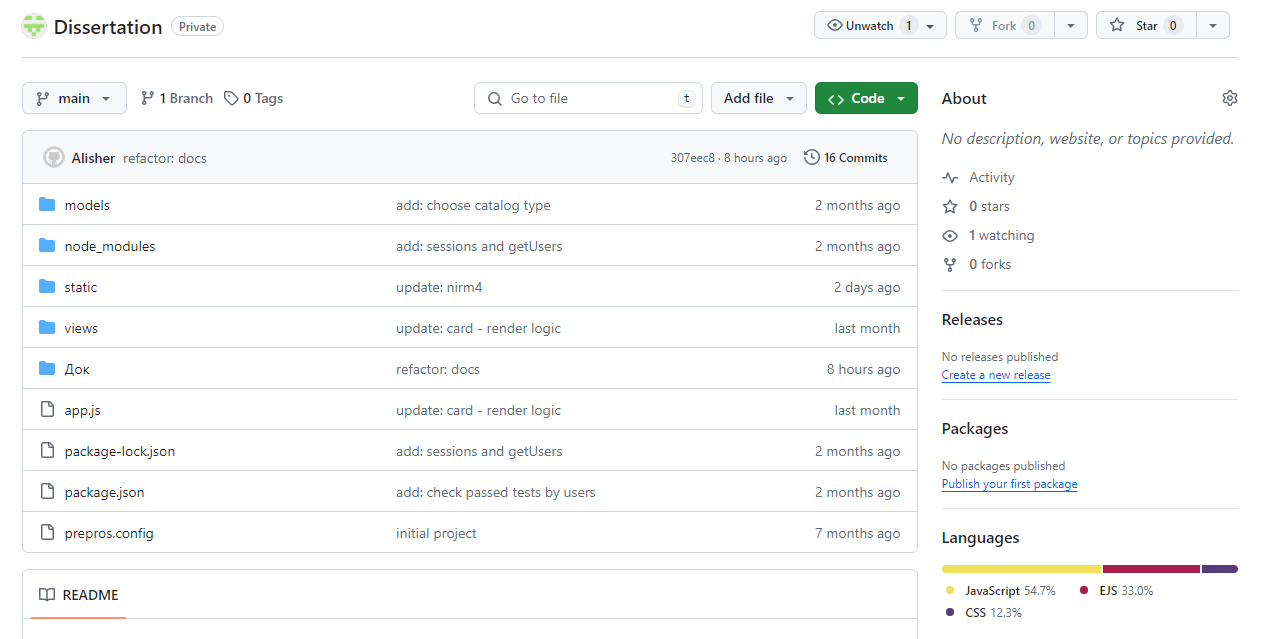


Рисунок 27. Страница GitHub

Далее подробно разберем функционал приложения, рассмотрим его ключевые возможности и специфические особенности, которые делают его уникальным и полезным для пользователей.



Рисунок 28. Функции получения и генерации списка пользователей

(только для админа)

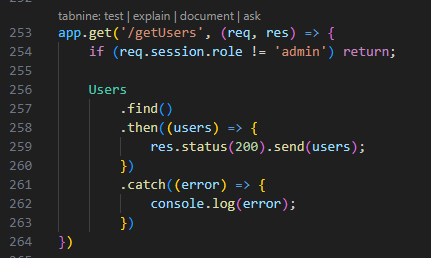


Рисунок 29. Возвращение списка пользователей



Рисунок 30. Создание и удаление пользователей

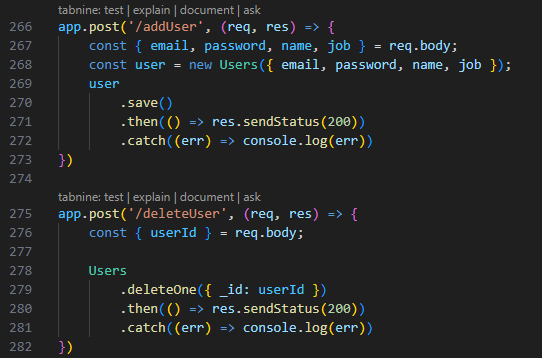


Рисунок 31. Обработка добавления и удаления пользователей

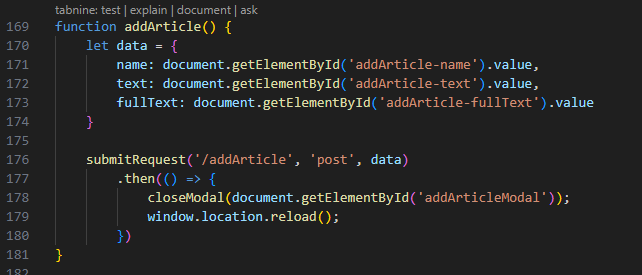


Рисунок 32. Функция создание новых статей

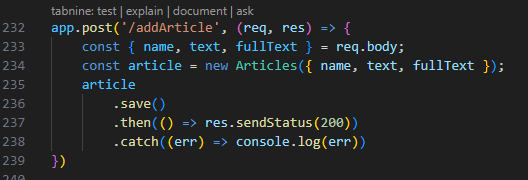


Рисунок 33. Добавление новых статей в БД

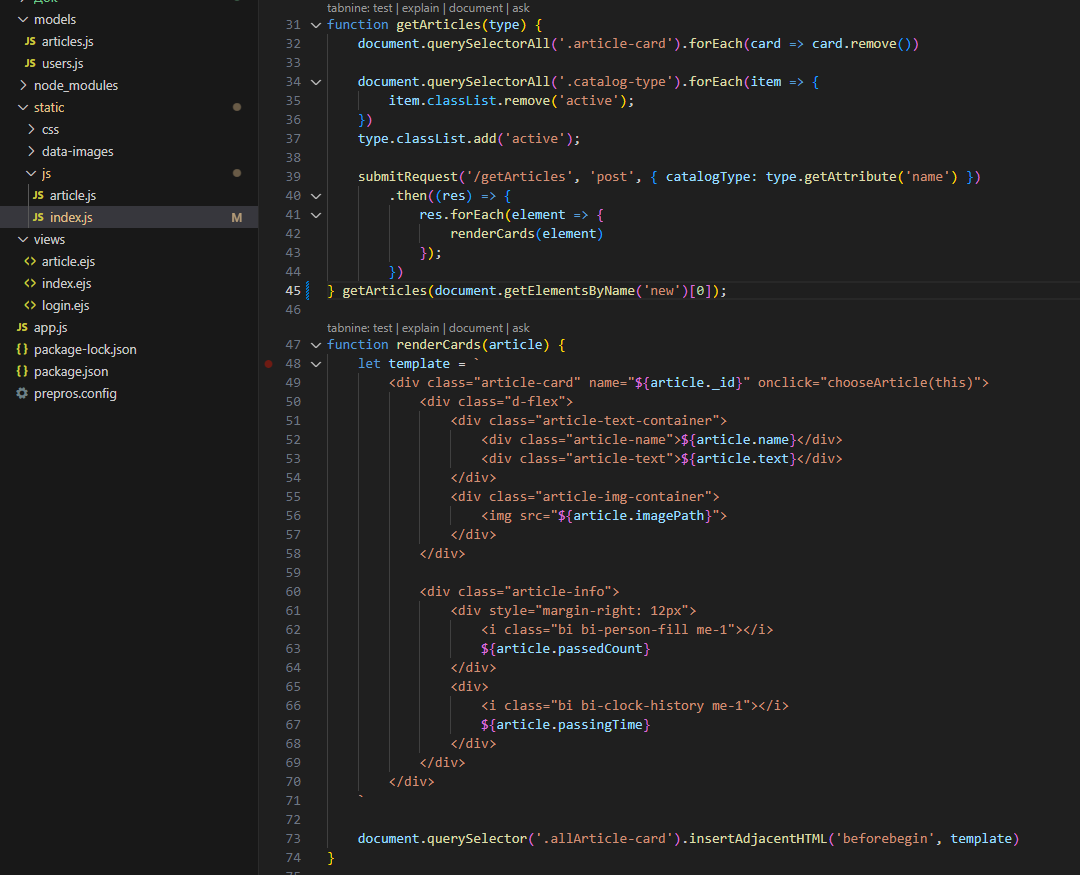


Рисунок 34. Функции получения и генерации карточек статей

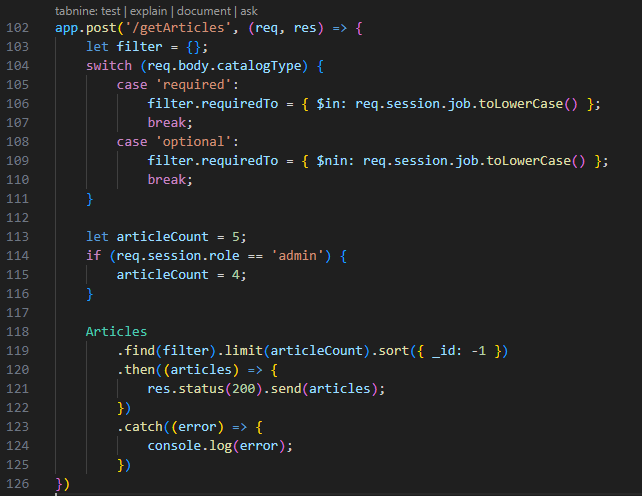


Рисунок 35. Обработка и отправка всех статей из БД

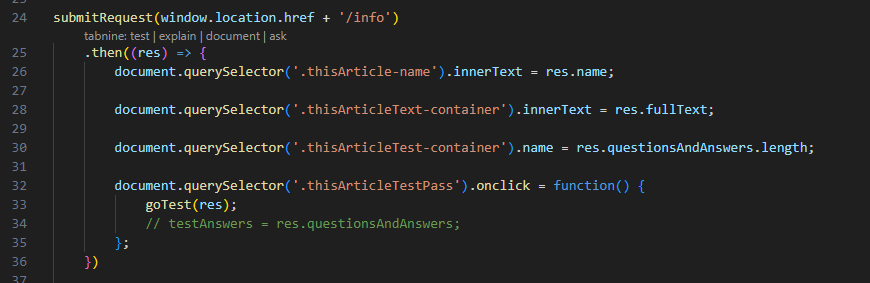


Рисунок 36. Получение и отображение содержимого статьи

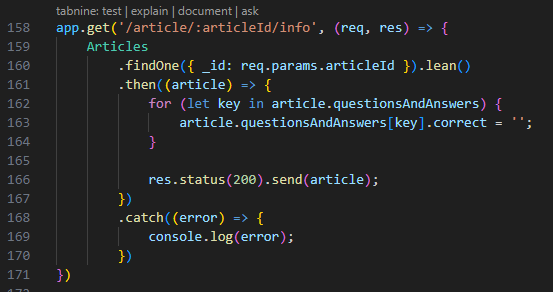


Рисунок 37. Отправка содержимого статьи клиенту



Рисунок 38. Функция генерации теста статьи



Рисунок 39. Функция создания теста для статьи

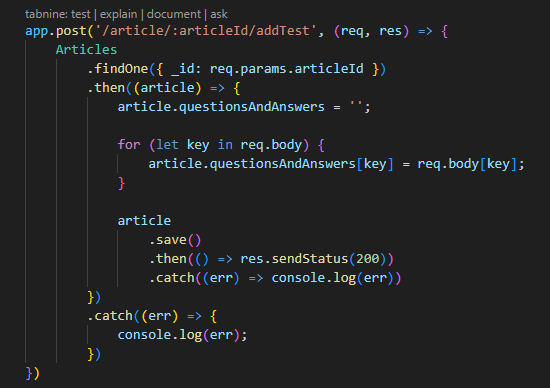


Рисунок 40. Сохранение теста в базу данных



Рисунок 41. Функция для отправки ответов

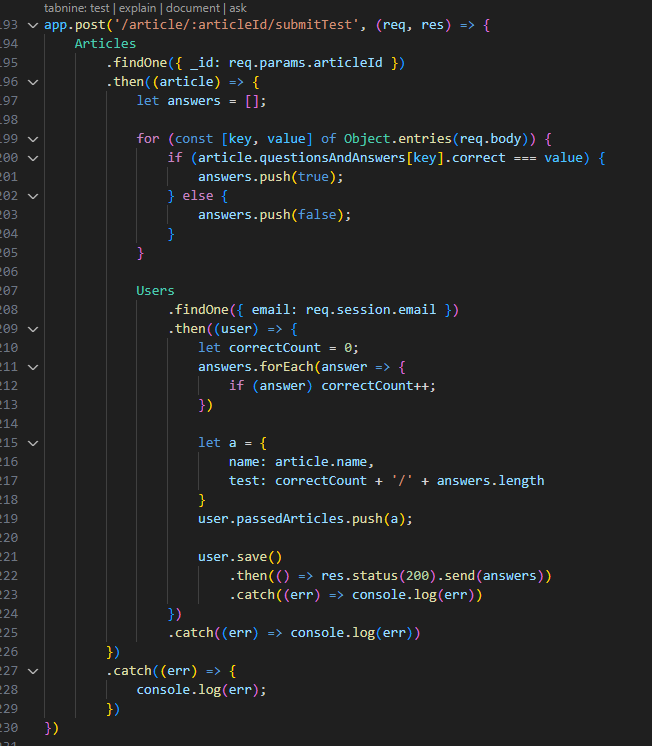


Рисунок 42. Подсчет количества верных ответов пользователя



Рисунок 43. Просмотр списка пройденных тестов пользователя

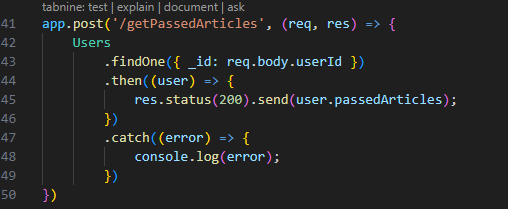


Рисунок 44. Отправка списка пройденных пользователем тестов

**3 ТЕСТИРОВАНИЕ И ВНЕДРЕНИЕ ПО**

Приложение может быть легко запущено на любом компьютере предприятия, подняв сервер локально. Для этого следует выполнить следующие шаги:

Установка зависимостей:

Убедиться, что на компьютере установлен Node.js. Если нет, его можно загрузить и установить с официального сайта Node.js.

Затем перейдя в корневую директорию проекта и выполните команду npm install в терминале, чтобы установить все необходимые зависимости, указанные в файле package.json.

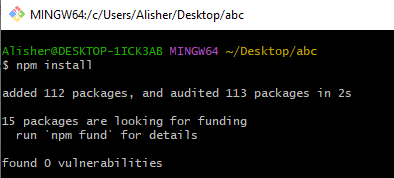


Рисунок 45. Установка зависимостей (модулей)

Запуск сервера:

После успешной установки зависимостей требуется запустить сервер Node.js, используя команду npm start или node app.js в терминале.

После запуска сервера он будет доступен по адресу http://localhost:3000

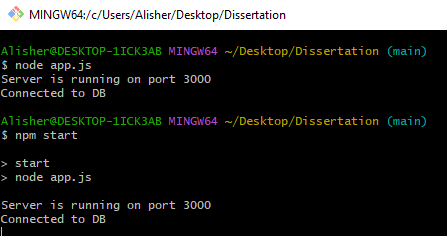


Рисунок 46. Запуск программного обеспечения

Доступ к приложению:

После запуска сервера любой сотрудник предприятия сможет получить доступ к приложению, введя веб-браузере IP-адрес ПК на котором был запущен сервер http://IP-адрес:3000

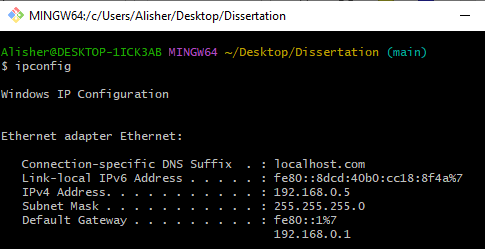


Рисунок 47. Команда для получения IP-адреса (windows)

Теперь приложение доступно локально на компьютерах сотрудников предприятия для использования в рабочих целях.

Эти простые шаги позволят любому предприятию быстро и легко запустить приложение локально на своем компьютере и начать использовать его в рабочей среде.

При первом доступе к приложению пользователь встречается с окном авторизации, где ему предоставляется возможность войти в систему, используя свои учетные данные.

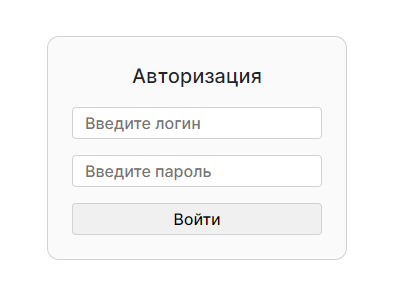


Рисунок 48. Окно авторизации

Главная страница приложения:

На главной странице пользователи встречаются с информацией о своем профиле, включая имя и должность.



Рисунок 49. Информация пользователя

Главным элементом страницы является каталог статей, представленных в виде удобных плиток. Каждая плитка содержит заголовок статьи, краткое описание, количество прочитавших, и оценку времени на прочтение. Пользователи имеют возможность сортировать статьи по разделам: "Новые", "Обязательные" (для пользователей с определенной должностью) и "Дополнительные" (необязательные).

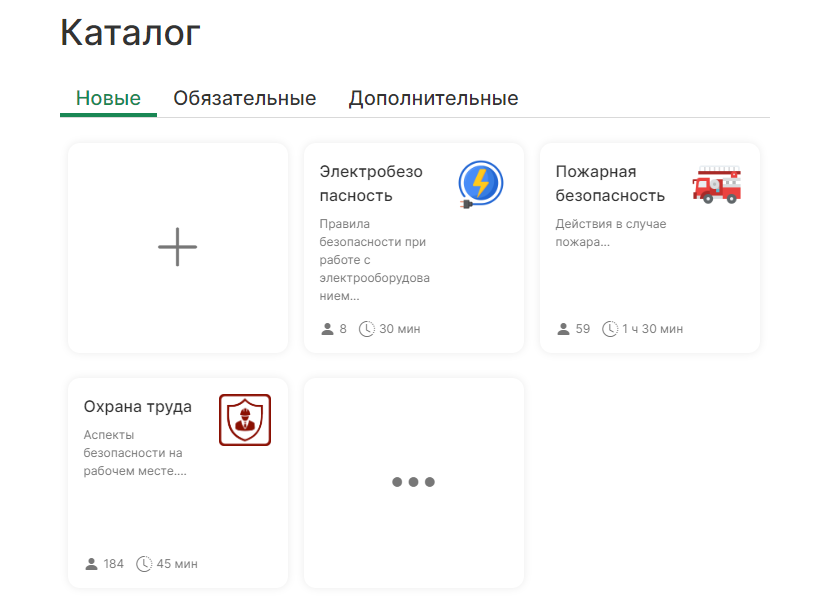


Рисунок 50. Каталог статей

Для администратора доступны дополнительные функции, такие как:

Добавление новых статей: Администратор может создавать новые статьи, используя специальную плитку на главной странице. При этом открывается модальное окно с полями для ввода необходимой информации.

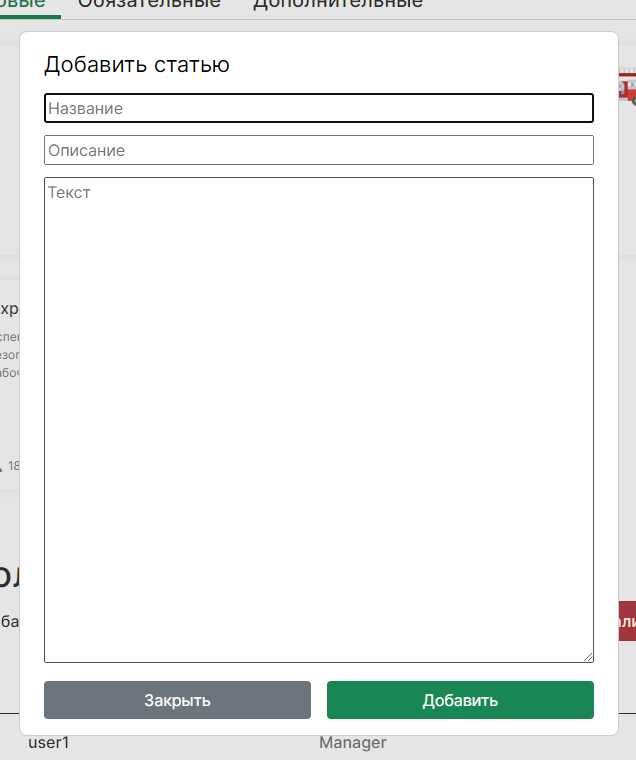


Рисунок 51. Форма создания статьи

Созданные статьи сразу можно увидеть в «Новые». Внутри карточек можно увидеть их название, краткое описания, количество вошедших в статью пользователей и приблизительное время прохождения.

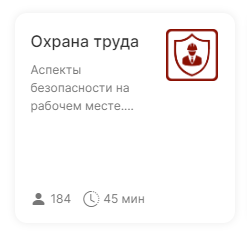


Рисунок 52. Карточка статьи

Управление пользователями: Также администратор видит список всех пользователей, включая их имена и должности. И имеет кнопки для добавления новых пользователей и удаления существующих.

Ниже секция показывающая список всех пользователей.

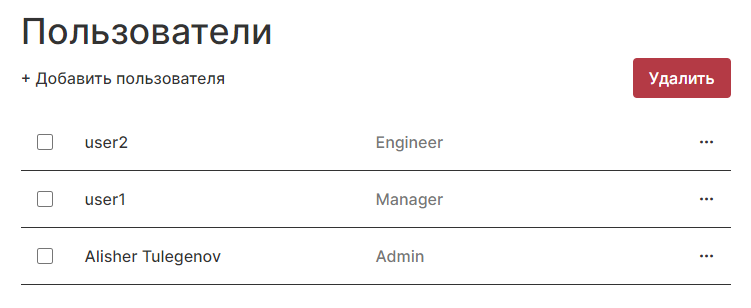


Рисунок 53. Блок взаимодействия с пользователями

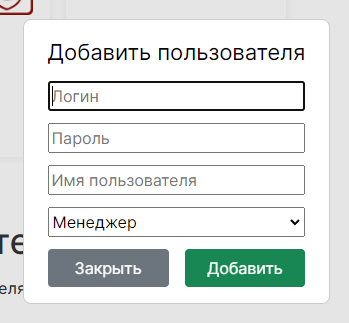


Рисунок 54. Окно создания новых пользователей

При переходе на страницу статьи, пользователь может видеть полное содержимое статьи, которое содержит полный текст и тест (если он там предусмотрен)

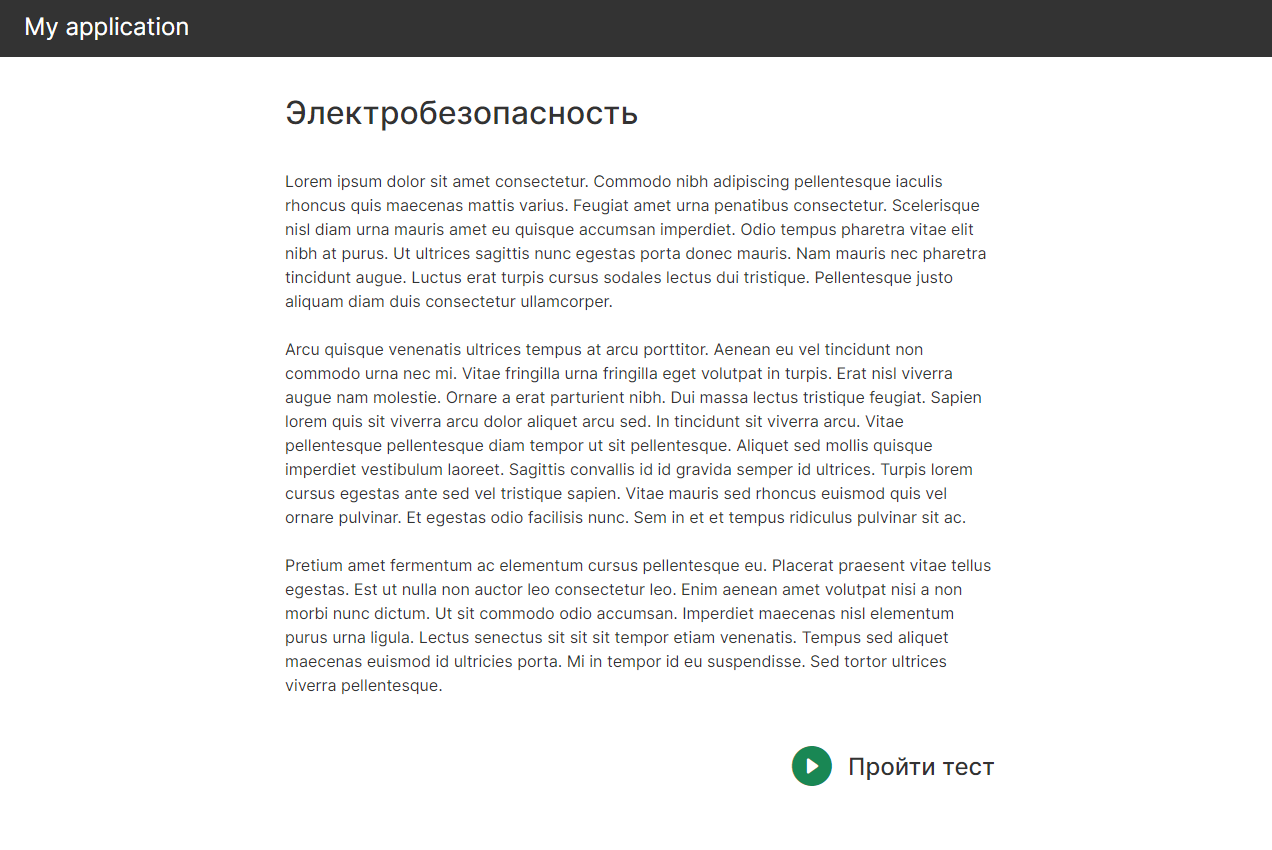


Рисунок 55. Страница статьи (пользователь)

Также у Админа присутствует кнопка для добавления теста. При добавлении теста, требуется заполнить поля с вопросами, ответами и указать правильный ответ.

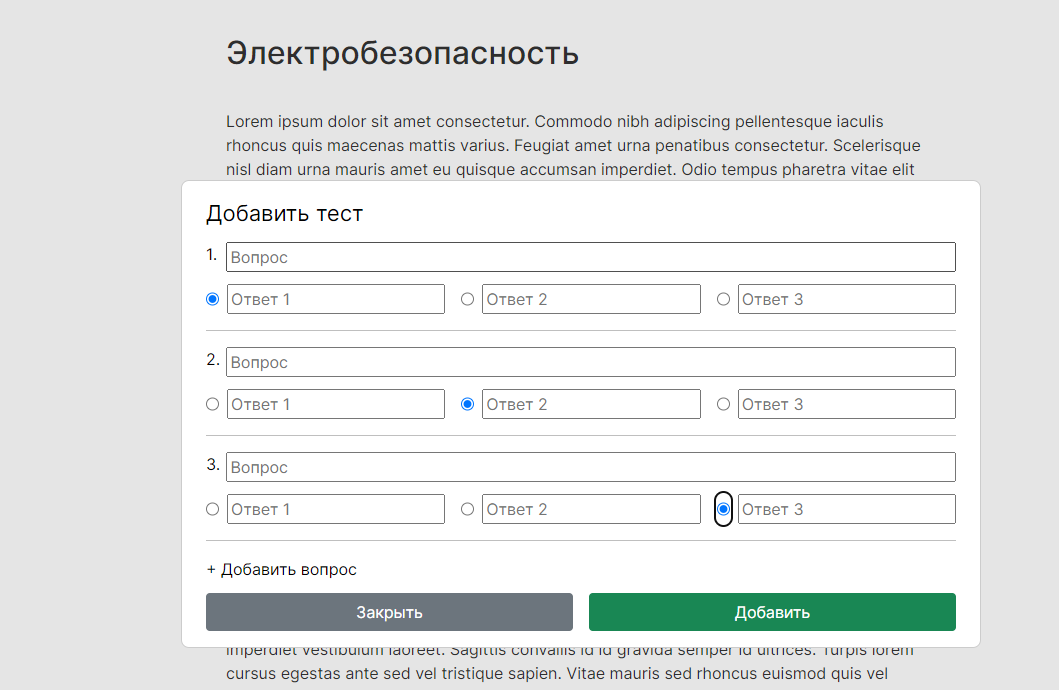


Рисунок 56. Форма создания тестов

Прохождение теста:

В случае, если к данной статье предусмотрен тест, пользователь перенаправляется на страницу с вопросами. После завершения теста, результаты (количество баллов) записываются в базу данных, а пользователь видит отчет о полученных баллах.

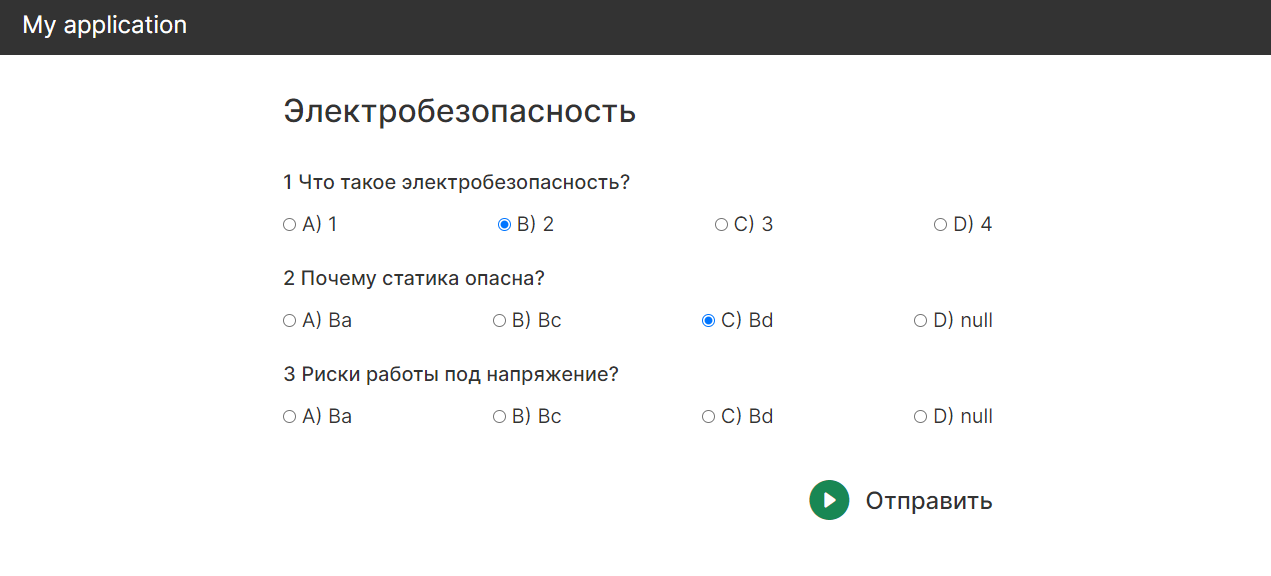


Рисунок 57. Страница прохождения теста

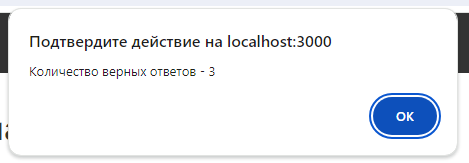


Рисунок 58. Получение ответа (оценки)

Так же если пользователь проходил тесты по статьям, администратор имеет возможность просматривать результаты тестирования в разделе «пользователи» нажав на три точки справа. Отчет включает в себя количество набранных и общих баллов.

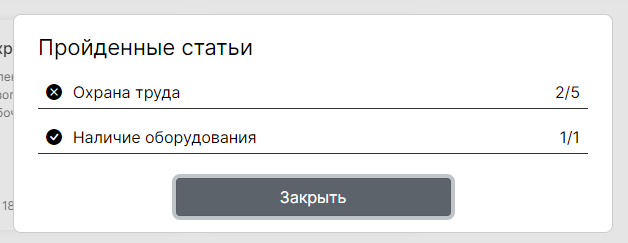


Рисунок 59. Окно пройденных статей

Это обеспечивает прозрачность и отслеживание процесса обучения сотрудников по технике безопасности.

Такой функционал позволяет пользователям эффективно проверять свои знания и предоставляет администратору инструменты для мониторинга успеваемости и обучения персонала.

Переход на страницы статьи и прохождение тестов:

После выбора статьи из каталога, пользователь попадает на страницу с развернутым текстовым содержанием статьи. Здесь ему предоставляется возможность ознакомиться с материалом более подробно. В нижней части страницы расположена кнопка, при нажатии на которую открывается доступ к тесту для проверки усвоенных знаний.

**ЗАКЛЮЧЕНИЕ**

В век развития технологий травматизм и профессиональные заболевания не редкость, поэтому охрана труда должна быть одной из основных задач руководства промышленного предприятия.

Разработка ПО для автоматизации ТБ является неотъемлемой частью современных промышленных процессов. Оно позволяет не только обеспечить безопасность и снизить риски, но и повысить эффективность, оптимизировать производственные процессы и улучшить общий стандарт безопасности в рабочей среде. При правильном подходе к разработке, внедрению и поддержке, такое ПО становится незаменимым инструментом, способствующим процветанию современных промышленных предприятий.

По этой причине стоит обратить внимание на инновационные решения в данной сфере, а именно на автоматизацию задач охраны труда.

**СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ**

1. Федоров А. В., Иванова Е. В. Автоматизация учета технических средств обеспечения безопасности: современные подходы и методы // Материалы X международной научно-практической конференции "Инновационные технологии в науке и образовании". - 2020.
2. Технологии автоматизации журнала техники безопасности: практический опыт и применение / Под ред. Иванова В. Н. - М.: Наука, 2021.
3. Джонсон Дж. Автоматизация процессов управления технической безопасностью на предприятии // Технологии безопасности. - 2019.
4. Техническая безопасность и автоматизация: сборник научных трудов / Под ред. Смирнова В. В. - СПб.: Издательство Политехнического университета, 2017.
5. https://blog.intalev.ru/post/avtomatizirovannyy-kontrol-tekhniki-bezopasnosti-v-stroitelstve/
6. Гринченко Ю. А., Соколов М. В. Применение информационных технологий для автоматизации учета и анализа технической безопасности // Вестник Костромского государственного университета им. Н. А. Некрасова. - 2018.
7. Алексеев А. В. Автоматизированные системы учета технической безопасности на производстве // Труды Международной научной конференции "Информационные технологии в науке, управлении, социальной сфере и медицине". - 2020.
8. Automated Safety Management Systems: A Literature Review // Center for Chemical Process Safety. - 2017.
9. Шабанов С. Г., Иванов В. Н., Ломовцев А. А. Применение программных систем для автоматизации учета технической безопасности // Технологии безопасности. - 2022.
10. Руководство по классическим приложениям [Электронный ресурс]. Режим доступа: https://learn.microsoft.com/ru-ru/dotnet/fundamentals/ (дата запроса: 08.02.2024).